ACL 2007

Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics

June 23–30, 2007
Prague, Czech Republic

The Association for Computational Linguistics

It is with great sadness that we record the death on 4 April 2007 of Karen Spärck Jones, who among many other honors and awards, was the President of ACL in 1994 and recipient of the ACL Lifetime Achievement Award in 2004.

These proceedings are dedicated to her memory.

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

Order copies of this and other ACL proceedings from:

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

Table of Contents

Preface: General Chair

On behalf of the organizing committee I am delighted to welcome you to the 45th Annual Meeting of the Association for Computational Linguistics, in Prague.

Setting up and running the ACL conference involves a lot of work by many people. Some of them are officially identified as being responsible for various aspects of the conference, while the contributions of others are less visible. I would like to say a warm thank you to the people named below, with apologies to anyone I have overlooked.

The Program Chairs, **Antal van den Bosch** and **Annie Zaenen** have done a great job in managing the almost 600 submissions for the main conference and putting together a high quality program. Through this process Antal has become a 'grandmaster' of the START paper management system and has given a lot of help to other chairs who have had to deal with their own sets of submissions. Many thanks also to their Area Chairs and the program committee of reviewers, and to **Florence Reeder** for coordinating the pre-submission mentoring service. (Antal and Annie reflect on their PC experience overleaf).

Sophia Ananiadou is Chair of the Demo/Poster part of the conference, and has overseen a separate review process to select a high quality set of presentations.

The Student Research Workshop Chairs, **Chris Biemann**, **Violeta Seretan** and **Ellen Riloff** have assembled an excellent program of papers and posters. I encourage everyone to attend the student workshop to hear about the exciting work being carried out by researchers just starting out on their careers in computational linguistics.

Workshops Chair **Simone Teufel** is overseeing 15 workshops – the most ever at an ACL conference – chosen (with the help of **Beth Ann Hockey**, **Katja Markert** and **Dekai Wu**) from a total of 27 proposals. The scale of the workshop program can be gauged by the fact that they received an aggregate total of 470 submissions (without even counting IWPT and EMNLP-CoNLL). **Joakim Nivre**, as Tutorials Chair, has assembled a program of 5 attractive and complementary tutorials, selected from 20 proposals with advice from **Walter Daelemans**, **Robert Dale**, **Nancy Ide**, **Diane Litman** and **Chris Manning**.

One of the most demanding yet least noticed organizational roles is that of Publications Chair. **Su Jian** has done a fantastic job in producing the hardcopy and electronic record of the conference, supported by his team—with notable contributions from **Upali Kohomban** who has cheerfully helped at all stages and whenever needed, day and night, weekdays and weekends.

Sponsorship is another success story, thanks to the Sponsorship Chairs **Martha Palmer**, **Gabor Proszeky**, **Jan Hajic** and **Jun'ichi Tsujii**, who have recruited 12 corporate sponsors. We are very grateful for their financial support.

Eva Hajicova, the Local Arrangements Chair, assisted by **Jan Hajic** and **Anna Kotesovcova**, have put in an enormous amount of detailed work to make this conference a success, ably supported by the local team of **Milan Fucik**, **Jaroslava Hlavacova**, **Marketa Lopatkova**, **Jiri Mirovsky**, **Pavel Pecina**,

Pavel Schlesinger, **Juraj Simlovic**, **Miroslav Spousta**, **Pavel Stranak**, **Zlatka Subrova**, **Jan Votrubec**, **Zdenek Zabokrtsky** and **Daniel Zeman**. From the ACL itself, **Kathy McCoy**, **Dragomir Radev**, **Priscilla Rasmussen**, **Mark Steedman** and **Jun'ichi Tsujii** have played strong roles in making decisions and giving advice, and kept everything on track while I was out of action due to ill health last year.

And finally, many thanks to the authors and presenters in the main conference, workshops and co-located events... and to all the participants. I hope you enjoy the formally organized aspects of the conference, take advantage of the opportunity to network with colleagues old and new, and that you also have a chance to appreciate the history and sights of Prague while you are here.

John Carroll
ACL 2007 General Chair

Preface: Program Chairs

The number of submissions for this ACL broke a new record: the program committee's selection of 131 papers was based on 588 submissions (after withdrawals). An updated program design with four parallel sessions and 25-minute papers allowed an acceptance rate of 22.3%, and an acceptance of all submissions that were recommended with priority by the area chairs.

First and foremost, we thank all the authors for submitting papers describing their recent work; the sheer amount of submissions reflects how active our field is. We thank **Florence Reeder** for provided mentoring to 17 author teams who felt they needed some writing support. For the selection, we are indebted to the 332 program committee members, who produced one to eleven reviews per reviewer, for a total of close to 1,800 reviews, and to the ten area chairs on whose shoulders rested most of the work of organizing the review process. We decided to work without an area chairs meeting: the two program co-chairs met for two days at Tilburg University, and interacted during that time vigorously with the area chairs by email and sometimes by phone.

As usual the main program will run for three days: there will be four parallel sessions of main session presentations, a demo/poster session organized by **Sophia Ananiadou**, nameMiroslav Spousta and **Zdenek Zabokrtsky**, and a Student Research Workshop – thanks to **Ellen Riloff**, **Violeta Seretan** and **Chris Biemann** for organizing it. Also as usual the conference is flanked by tutorial sessions and workshops; our thanks go to **Joakim Nivre** and **Simone Teufel** for organizing and compiling an excellent package.

The announcements of the ACL Lifetime Award and of the Best Paper Award will provide the customary suspense. They will take place in plenary sessions. Other plenary sessions will be devoted to the business meeting and the two invited talks, which this year will be delivered by **Tom Mitchell** and **Barney Pell**. We are grateful for their kind acceptation of our invitation.

We thank **John Carroll**, General Conference Chair, the Local Arrangements Committee headed by **Eva Hajicova**, and the ACL executive, especially **Dragomir Radev**, for their help and advice, and last year's co-chairs, **Claire Cardie** and **Pierre Isabelle**, for sharing their experience. Our sincere thanks go to **Su Jian** for putting together the proceedings.

Enjoy the conference,

Antal van den Bosch and Annie Zaenen
ACL-2007 Program Chairs

Organizers

General Chair:

John Carroll, University of Sussex, UK

Local Arrangements Chair:

Eva Hajicova, Charles University, Czech Republic

Program Chairs:

Antal van den Bosch, Tilburg University, The Netherlands
Annie Zaenen, Palo Alto Research Center (PARC), USA

Student Research Workshop:

Violeta Seretan, University of Geneva, Switzerland
Chris Biemann, University of Leipzig, Germany
Ellen Riloff (Faculty Advisor), University of Utah, USA

Workshop Chair:

Simone Teufel, University of Cambridge, UK

Tutorial Chair:

Joakim Nivre, Växjö University, Sweden

Demo/Poster Chair:

Sophia Ananiadou, The University of Manchester, UK

Exhibits Chairs:

Jaroslava Hlavacova, Charles University, Czech Republic
Pavel Pecina, Charles University, Czech Republic

Sponsorship Chairs:

Martha Palmer, University of Colorado, USA
Gabor Proszeky, Morphologic, Hungary
Jan Hajic, Charles University, Czech Republic
Jun'ichi Tsujii, University of Tokyo, Japan

Publicity Chairs:

Pavel Stranak, Charles University, Czech Republic
Jiri Mirovsky, Charles University, Czech Republic

Publication Chair:

Su Jian, Institute for Infocomm Research (I2R), Singapore

Mentoring Service:

Florence Reeder, The MITRE Corporation, USA

Student Volunteers:

Marketa Lopatkova, Charles University, Czech Republic

Webmasters:

Zlatka Subrova, Czech Republic
Juraj Simlovic, Czech Republic

Secretariat:

Anna Kotesovcova, Charles University, Czech Republic

Registration:

Priscilla Rasmussen, Association for Computational Linguistics (ACL)

ACL Executive Committee:

Mark Steedman, University of Edinburgh, UK
Bonnie Dorr, University of Maryland, USA
Steven Bird, University of Melbourne, Australia
Jun'ichi Tsujii, The University of Tokyo, Japan
Kathleen F. McCoy, University of Delaware, USA
Dragomir R. Radev, University of Michigan, USA
Owen Rambow, Columbia University, USA
Alex Lascarides, University of Edinburgh, UK
Keh-Yih Su, Behavior Design Corporation, Taiwan
Claire Cardie, Cornell University, USA
Nicoletta Calzolari, Istituto di Linguistica Computazionale del CNR, Italy

Program Committee

Program Chairs:

Antal van den Bosch, Tilburg University, The Netherlands
Annie Zaenen, Palo Alto Research Center, USA

Area Chairs:

Tim Baldwin, University of Melbourne, Australia
Kees van Deemter, University of Aberdeen, UK
Barbara Di Eugenio, University of Illinois at Chicago, USA
Josef van Genabith, Dublin City University, Ireland
Claire Grover, University of Edinburgh, UK
Diana McCarthy, University of Sussex, UK
Dan Roth, University of Illinois at Urbana-Champaign, USA
Richard Sproat, University of Illinois at Urbana-Champaign, USA
Marc Swerts, Tilburg University, The Netherlands
Andy Way, Dublin City University, Ireland

Program Committee Members:

Takeshi Abekawa, Eneko Agirre, Gregory Aist, Cyril Allauzen, Elisabeth André, Shlomo Argamon, Ron Artstein,

Collin Baker, Srinivas Bangalore, Colin Bannard, Regina Barzilay, Roberto Basili, John Bateman, Kenneth Beesley, Anja Belz, Slaven Bilac, Steven Bird, Guido Boella, Francis Bond, Kalina Bontcheva, Johan Bos, Gosse Bouma, Susan Brennan, Chris Brew, Ted Briscoe, Ralf Brown, Paul Buitelaar, Razvan Bunescu, Harry Bunt,

Aoife Cahill, Janet Cahn, Mary Elaine Califf, Charles Callaway, Chris Callison-Burch, Nicoletta Calzolari, Giuseppe Carenini, Michael Carl, Jean Carletta, Xavier Carreras, Justine Cassell, Joyce Chai, Jing-Shin Chang, Ming-Wei Chang, Jinying Chen, Keh-Jiann Chen, Colin Cherry, David Chiang, Grzegorz Chrupala, Jennifer Chu-Carroll, Ilyas Cicekli, Philipp Cimiano, Stephen Clark, Michael Collins, Michael Connor, Mark Core, Mathias Creutz, James Curran,

Walter Daelemans, Ido Dagan, Hercules Dalianis, Hal Daume, Eric de la Clergerie, Maarten de Rijke, Barbara Di Eugenio, Mona Diab, Gael Diaz, Bonnie Dorr, Laila Dybkjær,

Jason Eisner, Noemie Elhadad, Katrin Erk, Dominique Estival, Stefan Evert,

David Farwell, Afsaneh Fazly, Marcello Federico, Katja Filippova, Kate Forbes Riley, Mikel Forcada, George Foster, Mary Ellen Foster, Anette Frank, Reva Freedman, Guohong Fu, Pascale Fung,

Rob Gaizauskas, Jianfeng Gao, Kallirroi Georgila, Daniel Gildea, Roxana Girju, Alfio Gliozzo, Sharon Goldwater, Fernando Gomez, Nancy Green, Mark Greenwood, Ralph Grishman, Declan Groves,

Nizar Habash, Keith Hall, Susan Haller, Sanda Harabagiu, Henk Harkema, Tony Hartley, Sven Hartrumpf, Mary Hearne, Marti Hearst, Peter Heeman, James Henderson, Mark Hepple, Ryuichiro Higashinaka, Julia Hirschberg, Graeme Hirst, Barbora Hladka, Julia Hockenmaier, Deirdre Hogan, Baden Hughes,

Diana Inkpen, Kentaro Inui,

Martin Jansche, Valentin Jijkoun, Mark Johnson, Kristiina Jokinen, Pamela Jordan,

Laura Kallmeyer, Stephan Kanthak, Vangelis Karkaletsis, Daisuke Kawahara, John Kelleher, Frank Keller, Andre Kempe, Rodger Kibble, Adam Kilgarriff, Tracy King, Ewan Klein, Alexandre Klementiev, Kevin Knight, Alistair Knott, Philipp Koehn, Rob Koeling, Alexander Koller, Grzegorz Kondrak, Moshe Koppel, Valia Kordoni, Anna Korhonen, Emiel Krahmer, Sandra Kuebler, Jonas Kuhn, Sadao Kurohashi,

Philippe Langlais, Guy Lapalme, Mirella Lapata, Staffan Larsson, Alberto Lavelli, Alon Lavie, Lillian Lee, Jochen Leidner, Oliver Lemon, Yves Lepage, Leonardo Lesmo, Gina Levow, Ian Lewin, William Lewis, Mu Li, Dekang Lin, Jimmy Lin, Diane Litman, Ting Liu, Saturnino Luz,

Bernardo Magnini, Gideon Mann, Chris Manning, Daniel Marcu, Katja Markert, Lluis Marquez, Erwin Marsi, David Martinez, Carlos Martin-Vide, Evgeny Matusov, John Maxwell, Mike Maxwell, Diana Maynard, Andrew McCallum, Ryan McDonald, Kathy McKeown, Susan McRoy, Mike McTear, Dan Melamed, Chris Mellish, Arul Menezes, Helen Meng, Paola Merlo, Detmar Meurers, Rada Mihalcea, Maria Milosavljevic, Eleni Miltsakaki, David Milward, Yusuke Miyao, Dan Moldovan, Diego Molla, Raymond Mooney, Roger Moore, Alessandro Moschitti, Karin Mueller, Reinhard Muskens,

Hiroshi Nakagawa, Hiromi Nakaiwa, Roberto Navigli, Mark-Jan Nederhof, Goran Nenadic, John Nerbonne, Hwee Tou Ng, Vincent Ng, Jian-Yun Nie, Malvina Nissim, Joakim Nivre, David Novick,

Jon Oberlander, Franz Och, Stephan Oepen, Kemal Oflazer, Ian O'Neill, Miles Osborne, Mari Ostendorf,

Tim Paek, Martha Palmer, Patrick Pantel, Ivandre Paraboni, Becky Passonneau, Michael Paul, Fernando Pereira, Scott Piao, Manfred Pinkal, Paul Piwek, Massimo Poesio, Joe Polifroni, Richard Power, John Prager, Stephen Pulman, Vasin Punyakanok,

Chris Quirk,

Reinhard Rapp, Lev Ratinov, Ehud Reiter, Martin Reynaert, Stefan Riezler, German Rigau, Ellen Riloff, Brian Roark, Laurent Romary,

Bogdan Sacaleanu, Horacio Saggion, Anna Sagvall Hein, Mark Sanderson, Anoop Sarkar, Avik Sarkar, Helmut Schmid, Gerold Schneider, Patrick Schone, Hinrich Schuetze, Sabine Schulte im Walde, Satoshi Sekine, Stephanie Seneff, Izhak Shafran, Advaith Siddharthan, Khalil Simaan, Kevin Small, Noah A. Smith, Harold Somers, Virach Sornlertlamvanich, Manfred Stede, Mark Steedman, Amanda Stent, Mark Stevenson, Carlo Strapparava, Oliver Streiter, Nicolas Stroppa, Michael Strube, Jian Su, Rajen Subba, Stan Szpakowicz,

Maite Taboada, Hiroya Takamura, Kumiko Tanaka-Ishii, Louis ten Bosch, Joel Tetreault, Simone Teufel, Mariet Theune, Joerg Tiedemann, Takenobu Tokunaga, Kristina Toutanova, Isabel Trancoso, Richard Tsai, Gokhan Tur,

Nicola Ueffing, Nathan Vaillette, Gertjan van Noord, Menno van Zaanen, Sebastian Varges, Paola Velardi, Renata Vieira, Begona Villada Moiron, Carl Vogel, Stephan Vogel, Piek Vossen,

Marilyn Walker, Haifeng Wang, Xinglong Wang, Bonnie Webber, David Weir, Ben Wellner, Richard Wicentowski, Janyce Wiebe, Ross Wilkinson, Yorick Wilks, Theresa Wilson, Shuly Wintner, Dekai Wu,

Fei Xia, Nianwen Xue, Endong Xun,

Muyun Yang, Scott Wen-tau Yih, Anssi Yli-Jyrä,

Dmitry Zelenko, Richard Zens, ChengXiang Zhai, Min Zhang, Tong Zhang, Guodong Zhou

Conference Program

Monday, June 25, 2007

8:45–9:00 Opening Session

9:00–10:00 Invited Talk by Tom Mitchell

10:00–10:30 Break

Session 1A: Machine Translation 1

10:30–10:55 *Guiding Statistical Word Alignment Models With Prior Knowledge*
Yonggang Deng and Yuqing Gao

10:55–11:20 *A Discriminative Syntactic Word Order Model for Machine Translation*
Pi-Chuan Chang and Kristina Toutanova

11:20–11:45 *Tailoring Word Alignments to Syntactic Machine Translation*
John DeNero and Dan Klein

11:45–12:10 *Transductive Learning for Statistical Machine Translation*
Nicola Ueffing, Gholamreza Haffari and Anoop Sarkar

Session 1B: Word Sense Disambiguation

10:30–10:55 *Word Sense Disambiguation Improves Statistical Machine Translation*
Yee Seng Chan, Hwee Tou Ng and David Chiang

10:55–11:20 *Learning Expressive Models for Word Sense Disambiguation*
Lucia Specia, Mark Stevenson and Maria das Graças Volpe Nunes

11:20–11:45 *Domain Adaptation with Active Learning for Word Sense Disambiguation*
Yee Seng Chan and Hwee Tou Ng

11:45–12:10 *Making Lexical Ontologies Functional and Context-Sensitive*
Tony Veale and Yanfen Hao

Monday, June 25, 2007 (continued)

Session 1C: Language Modeling 1

10:30–10:55 *A Bayesian Model for Discovering Typological Implications*
Hal Daumé III and Lyle Campbell

10:55–11:20 *A Discriminative Language Model with Pseudo-negative Samples*
Daisuke Okanohara and Jun'ichi Tsujii

11:20–11:45 *Detecting Erroneous Sentences using Automatically Mined Sequential Patterns*
Guihua Sun, Xiaohua Liu, Gao Cong, Ming Zhou, Zhongyang Xiong, John Lee and Chin-Yew Lin

11:45–12:10 *Vocabulary Decomposition for Estonian Open Vocabulary Speech Recognition*
Antti Puurula and Mikko Kurimo

Session 1D: Phonology and Morphology 1

10:30–10:55 *Phonological Constraints and Morphological Preprocessing for Grapheme-to-Phoneme Conversion*
Vera Demberg, Helmut Schmid and Gregor Möhler

10:55–11:20 *Redundancy Ratio: An Invariant Property of the Consonant Inventories of the World's Languages*
Animesh Mukherjee, Monojit Choudhury, Anupam Basu and Niloy Ganguly

11:20–11:45 *Multilingual Transliteration Using Feature based Phonetic Method*
Su-Youn Yoon, Kyoung-Young Kim and Richard Sproat

11:45–12:10 *Semantic Transliteration of Personal Names*
Haizhou Li, Khe Chai Sim, Jin-Shea Kuo and Minghui Dong

12:10–13:30 Lunch

Session 2A: Machine Translation 2

13:30–13:55 *Generating Complex Morphology for Machine Translation*
Einat Minkov, Kristina Toutanova and Hisami Suzuki

13:55–14:20 *Assisting Translators in Indirect Lexical Transfer*
Bogdan Babych, Anthony Hartley, Serge Sharoff and Olga Mudraya

Monday, June 25, 2007 (continued)

14:20–14:45 *Forest Rescoring: Faster Decoding with Integrated Language Models*
Liang Huang and David Chiang

14:45–15:10 *Statistical Machine Translation through Global Lexical Selection and Sentence Reconstruction*
Srinivas Bangalore, Patrick Haffner and Stephan Kanthak

Session 2B: Grammars

13:30–13:55 *Mildly Context-Sensitive Dependency Languages*
Marco Kuhlmann and Mathias Möhl

13:55–14:20 *Transforming Projective Bilexical Dependency Grammars into Efficiently-parsable CFGs with Unfold-Fold*
Mark Johnson

14:20–14:45 *Parsing and Generation as Datalog Queries*
Makoto Kanazawa

14:45–15:10 *Optimizing Grammars for Minimum Dependency Length*
Daniel Gildea and David Temperley

Session 2C: Semantic Role Labeling

13:30–13:55 *Generalizing Semantic Role Annotations Across Syntactically Similar Verbs*
Andrew Gordon and Reid Swanson

13:55–14:20 *A Grammar-driven Convolution Tree Kernel for Semantic Role Classification*
Min Zhang, Wanxiang Che, Aiti Aw, Chew Lim Tan, Guodong Zhou, Ting Liu and Sheng Li

14:20–14:45 *Learning Predictive Structures for Semantic Role Labeling of NomBank*
Chang Liu and Hwee Tou Ng

14:45–15:10 *A Simple, Similarity-based Model for Selectional Preferences*
Katrin Erk

Monday, June 25, 2007 (continued)

Session 2D: Language Resources

13:30–13:55 *SVM Model Tampering and Anchored Learning: A Case Study in Hebrew NP Chunking*
Yoav Goldberg and Michael Elhadad

13:55–14:20 *Fully Unsupervised Discovery of Concept-Specific Relationships by Web Mining*
Dmitry Davidov, Ari Rappoport and Moshe Koppel

14:20–14:45 *Adding Noun Phrase Structure to the Penn Treebank*
David Vadas and James Curran

14:45–15:10 *Formalism-Independent Parser Evaluation with CCG and DepBank*
Stephen Clark and James Curran

15:10–15:45 Break

Session 3A, Machine Learning Methods 1

15:45–16:10 *Frustratingly Easy Domain Adaptation*
Hal Daumé III

16:10–16:35 *Instance Weighting for Domain Adaptation in NLP*
Jing Jiang and ChengXiang Zhai

16:35–17:00 *The Infinite Tree*
Jenny Rose Finkel, Trond Grenager and Christopher D. Manning

17:00–17:25 *Guiding Semi-Supervision with Constraint-Driven Learning*
Ming-Wei Chang, Lev Ratinov and Dan Roth

Session 3B: Machine Translation 3

15:45–16:10 *Supertagged Phrase-Based Statistical Machine Translation*
Hany Hassan, Khalil Sima'an and Andy Way

16:10–16:35 *Regression for Sentence-Level MT Evaluation with Pseudo References*
Joshua S. Albrecht and Rebecca Hwa

16:35–17:00 *Bootstrapping Word Alignment via Word Packing*
Yanjun Ma, Nicolas Stroppa and Andy Way

17:00–17:25 *Improved Word-Level System Combination for Machine Translation*
Antti-Veikko Rosti, Spyros Matsoukas and Richard Schwartz

Monday, June 25, 2007 (continued)

Session 3C: Generation

15:45–16:10 *Generating Constituent Order in German Clauses*
Katja Filippova and Michael Strube

16:10–16:35 *A Symbolic Approach to Near-Deterministic Surface Realisation using Tree Adjoining Grammar*
Claire Gardent and Eric Kow

16:35–17:00 *Sentence Generation as a Planning Problem*
Alexander Koller and Matthew Stone

17:00–17:25 *GLEU: Automatic Evaluation of Sentence-Level Fluency*
Andrew Mutton, Mark Dras, Stephen Wan and Robert Dale

Session 3D: Multimodality 1

15:45–16:10 *Conditional Modality Fusion for Coreference Resolution*
Jacob Eisenstein and Randall Davis

16:10–16:35 *The Utility of a Graphical Representation of Discourse Structure in Spoken Dialogue Systems*
Mihai Rotaru and Diane Litman

16:35–17:00 *Automated Vocabulary Acquisition and Interpretation in Multimodal Conversational Systems*
Yi Liu, Joyce Chai and Rong Jin

17:00–17:25 *A Multimodal Interface for Access to Content in the Home*
Michael Johnston, Luis Fernando D'Haro, Michelle Levine and Bernard Renger

Tuesday, June 26, 2007

Session 4A, Parsing 1

09:00–09:25 *Fast Unsupervised Incremental Parsing*
Yoav Seginer

09:25–09:50 *K-best Spanning Tree Parsing*
Keith Hall

09:50–10:15 *Is the End of Supervised Parsing in Sight?*
Rens Bod

10:15–10:40 *An Ensemble Method for Selection of High Quality Parses*
Roi Reichart and Ari Rappoport

Session 4B: Sentiment 1

09:00–09:25 *Opinion Mining using Econometrics: A Case Study on Reputation Systems*
Anindya Ghose, Panagiotis Ipeirotis and Arun Sundararajan

09:25–09:50 *PageRanking WordNet Synsets: An Application to Opinion Mining*
Andrea Esuli and Fabrizio Sebastiani

09:50–10:15 *Structured Models for Fine-to-Coarse Sentiment Analysis*
Ryan McDonald, Kerry Hannan, Tyler Neylon, Mike Wells and Jeff Reynar

10:15–10:40 *Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification*
John Blitzer, Mark Dredze and Fernando Pereira

Session 4C: Paraphrasing, Textual Entailment

09:00–09:25 *Clustering Clauses for High-Level Relation Detection: An Information-theoretic Approach*
Samuel Brody

09:25–09:50 *Instance-based Evaluation of Entailment Rule Acquisition*
Idan Szpektor, Eyal Shnarch and Ido Dagan

09:50–10:15 *Statistical Machine Translation for Query Expansion in Answer Retrieval*
Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal and Yi Liu

10:15–10:40 *A Computational Model of Text Reuse in Ancient Literary Texts*
John Lee

Tuesday, June 26, 2007 (continued)

Session 4D: Discourse and Dialog 1

09:00–09:25 *Finding Document Topics for Improving Topic Segmentation*
Olivier Ferret

09:25–09:50 *The Utility of Parse-derived Features for Automatic Discourse Segmentation*
Seeger Fisher and Brian Roark

09:50–10:15 *PERSONAGE: Personality Generation for Dialogue*
François Mairesse and Marilyn Walker

10:15–10:40 *Making Sense of Sound: Unsupervised Topic Segmentation over Acoustic Input*
Igor Malioutov, Alex Park, Regina Barzilay and James Glass

10:40–11:10 Break

11:10-12:10 LifeTime Achievement Award

12:10–13:30 Lunch

13:00-14:30 ACL Business Meeting

Session 5A: Language Modeling 2

14:30–14:55 *Randomised Language Modelling for Statistical Machine Translation*
David Talbot and Miles Osborne

14:55–15:20 *Bilingual-LSA Based LM Adaptation for Spoken Language Translation*
Yik-Cheung Tam, Ian Lane and Tanja Schultz

Session 5B: Coreference

14:30–14:55 *Coreference Resolution Using Semantic Relatedness Information from Automatically Discovered Patterns*
Xiaofeng Yang and Jian Su

14:55–15:20 *Semantic Class Induction and Coreference Resolution*
Vincent Ng

Tuesday, June 26, 2007 (continued)

Session 5C: Summarization

14:30–14:55 *Generating a Table-of-Contents*
S. R. K. Branavan, Pawan Deshpande and Regina Barzilay

14:55–15:20 *Towards an Iterative Reinforcement Approach for Simultaneous Document Summarization and Keyword Extraction*
Xiaojun Wan, Jianwu Yang and Jianguo Xiao

Session 5D: Semantic Relations

14:30–14:55 *Fast Semantic Extraction Using a Novel Neural Network Architecture*
Ronan Collobert and Jason Weston

14:55–15:20 *Improving the Interpretation of Noun Phrases with Cross-linguistic Information*
Roxana Girju

15:20–15:45 Break

Session 6A: Information Extraction

15:45–16:10 *Learning to Extract Relations from the Web using Minimal Supervision*
Razvan Bunescu and Raymond Mooney

16:10–16:35 *A Seed-driven Bottom-up Machine Learning Framework for Extracting Relations of Various Complexity*
Feiyu Xu, Hans Uszkoreit and Hong Li

16:35–17:00 *A Multi-resolution Framework for Information Extraction from Free Text*
Mstislav Maslennikov and Tat-Seng Chua

17:00–17:25 *Using Corpus Statistics on Entities to Improve Semi-supervised Relation Extraction from the Web*
Benjamin Rosenfeld and Ronen Feldman

Tuesday, June 26, 2007 (continued)

Session 6B: Parsing 2

15:45–16:10 *Beyond Projectivity: Multilingual Evaluation of Constraints and Measures on Non-Projective Structures*
Jiří Havelka

16:10–16:35 *Self-Training for Enhancement and Domain Adaptation of Statistical Parsers Trained on Small Datasets*
Roi Reichart and Ari Rappoport

16:35–17:00 *HPSG Parsing with Shallow Dependency Constraints*
Kenji Sagae, Yusuke Miyao and Jun'ichi Tsujii

17:00–17:25 *Constituent Parsing with Incremental Sigmoid Belief Networks*
Ivan Titov and James Henderson

Session 6C: Multilinguality 1

15:45–16:10 *Corpus Effects on the Evaluation of Automated Transliteration Systems*
Sarvnaz Karimi, Andrew Turpin and Falk Scholer

16:10–16:35 *Collapsed Consonant and Vowel Models: New Approaches for English-Persian Transliteration and Back-Transliteration*
Sarvnaz Karimi, Falk Scholer and Andrew Turpin

16:35–17:00 *Alignment-Based Discriminative String Similarity*
Shane Bergsma and Grzegorz Kondrak

17:00–17:25 *Bilingual Terminology Mining - Using Brain, not brawn comparable corpora*
Emmanuel Morin, Béatrice Daille, Koichi Takeuchi and Kyo Kageura

Session 6D: Language Modeling 3

15:45–16:10 *Unsupervised Language Model Adaptation Incorporating Named Entity Information*
Feifan Liu and Yang Liu

16:10–16:35 *Coordinate Noun Phrase Disambiguation in a Generative Parsing Model*
Deirdre Hogan

16:35–17:00 *A Unified Tagging Approach to Text Normalization*
Conghui Zhu, Jie Tang, Hang Li, Hwee Tou Ng and Tiejun Zhao

17:00–17:25 *Sparse Information Extraction: Unsupervised Language Models to the Rescue*
Doug Downey, Stefan Schoenmackers and Oren Etzioni

Wednesday, June 27, 2007

Session 7A: Machine Translation 4

09:00–09:25 *Forest-to-String Statistical Translation Rules*
Yang Liu, Yun Huang, Qun Liu and Shouxun Lin

09:25–09:50 *Ordering Phrases with Function Words*
Hendra Setiawan, Min-Yen Kan and Haizhou Li

09:50–10:15 *A Probabilistic Approach to Syntax-based Reordering for Statistical Machine Translation*
Chi-Ho Li, Minghui Li, Dongdong Zhang, Mu Li, Ming Zhou and Yi Guan

10:15–10:40 *Machine Translation by Triangulation: Making Effective Use of Multi-Parallel Corpora*
Trevor Cohn and Mirella Lapata

Session 7B: Sequence Processing

09:00–09:25 *A Maximum Expected Utility Framework for Binary Sequence Labeling*
Martin Jansche

09:25–09:50 *A Fully Bayesian Approach to Unsupervised Part-of-speech Tagging*
Sharon Goldwater and Tom Griffiths

09:50–10:15 *Computationally Efficient M-Estimation of Log-Linear Structure Models*
Noah A. Smith, Douglas L. Vail and John D. Lafferty

10:15–10:40 *Guided Learning for Bidirectional Sequence Classification*
Libin Shen, Giorgio Satta and Aravind Joshi

Session 7C: Question Answering

09:00–09:25 *Different Structures for Evaluating Answers to Complex Questions: Pyramids Won't Topple, and Neither Will Human Assessors*
Hoa Trang Dang and Jimmy Lin

09:25–09:50 *Exploiting Syntactic and Shallow Semantic Kernels for Question Answer Classification*
Alessandro Moschitti, Silvia Quarteroni, Roberto Basili and Suresh Manandhar

09:50–10:15 *Language-independent Probabilistic Answer Ranking for Question Answering*
Jeongwoo Ko, Teruko Mitamura and Eric Nyberg

Wednesday, June 27, 2007 (continued)

Session 7D: Discourse and Dialog 2

09:00–09:25 *Learning to Compose Effective Strategies from a Library of Dialogue Components*
Martijn Spitters, Marco De Boni, Jakub Zavrel and Remko Bonnema

09:25–09:50 *On the Role of Context and Prosody in the Interpretation of 'okay'*
Agustín Gravano, Stefan Benus, Héctor Chávez, Julia Hirschberg and Lauren Wilcox

09:50–10:15 *Predicting Success in Dialogue*
David Reitter and Johanna D. Moore

10:15–10:40 *Resolving It, This, and That in Unrestricted Multi-Party Dialog*
Christoph Müller

10:40–11:10 Break

11:10–12:10 Invited Talk by Barney Pell

12:10–13:30 Lunch

Session 8A: Machine Learning Methods 2

13:30–13:55 *A Comparative Study of Parameter Estimation Methods for Statistical Natural Language Processing*
Jianfeng Gao, Galen Andrew, Mark Johnson and Kristina Toutanova

13:55–14:20 *Grammar Approximation by Representative Sublanguage: A New Model for Language Learning*
Smaranda Muresan and Owen Rambow

14:20–14:45 *Chinese Segmentation with a Word-Based Perceptron Algorithm*
Yue Zhang and Stephen Clark

14:45–15:10 *Unsupervised Coreference Resolution in a Nonparametric Bayesian Model*
Aria Haghighi and Dan Klein

Wednesday, June 27, 2007 (continued)

Session 8B: Machine Translation and Multilinguality

13:30–13:55 *Pivot Language Approach for Phrase-Based Statistical Machine Translation*
Hua Wu and Haifeng Wang

13:55–14:20 *Bootstrapping a Stochastic Transducer for Arabic-English Transliteration Extraction*
Tarek Sherif and Grzegorz Kondrak

14:20–14:45 *Benefits of the Passively Parallel Rosetta Stone? Cross-Language Information Retrieval with over 30 Languages*
Peter A. Chew and Ahmed Abdelali

14:45–15:10 *A Re-examination of Machine Learning Approaches for Sentence-Level MT Evaluation*
Joshua S. Albrecht and Rebecca Hwa

Session 8C: Lexicon and Lexical Semantics

13:30–13:55 *Automatic Acquisition of Ranked Qualia Structures from the Web*
Philipp Cimiano and Johanna Wenderoth

13:55–14:20 *A Sequencing Model for Situation Entity Classification*
Alexis Palmer, Elias Ponvert, Jason Baldridge and Carlota Smith

14:20–14:45 *Words and Echoes: Assessing and Mitigating the Non-Randomness Problem in Word Frequency Distribution Modeling*
Marco Baroni and Stefan Evert

14:45–15:10 *A System for Large-Scale Acquisition of Verbal, Nominal and Adjectival Subcategorization Frames from Corpora*
Judita Preiss, Ted Briscoe and Anna Korhonen

Wednesday, June 27, 2007 (continued)

Session 8D: Phonology and Morphology 2

13:30–13:55 *A Language-Independent Unsupervised Model for Morphological Segmentation*
Vera Demberg

13:55–14:20 *Using Mazurkiewicz Trace Languages for Partition-Based Morphology*
François Barthélemy

14:20–14:45 *Much ado about nothing: A Social Network Model of Russian Paradigmatic Gaps*
Robert Daland, Andrea D. Sims and Janet Pierrehumbert

14:45–15:10 *Substring-Based Transliteration*
Tarek Sherif and Grzegorz Kondrak

15:10–15:45 Break

Session 9A: Parsing 3

15:45–16:10 *Pipeline Iteration*
Kristy Hollingshead and Brian Roark

16:10–16:35 *Learning Synchronous Grammars for Semantic Parsing with Lambda Calculus*
Yuk Wah Wong and Raymond Mooney

16:35–17:00 *Generalizing Tree Transformations for Inductive Dependency Parsing*
Jens Nilsson, Joakim Nivre and Johan Hall

Session 9B: Sentiment 2

15:45–16:10 *Learning Multilingual Subjective Language via Cross-Lingual Projections*
Rada Mihalcea, Carmen Banea and Janyce Wiebe

16:10–16:35 *Sentiment Polarity Identification in Financial News: A Cohesion-based Approach*
Ann Devitt and Khurshid Ahmad

16:35–17:00 *Weakly Supervised Learning for Hedge Classification in Scientific Literature*
Ben Medlock and Ted Briscoe

Wednesday, June 27, 2007 (continued)

Session 9C: Multimodality 2

15:45–16:10 *Text Analysis for Automatic Image Annotation*
Koen Deschacht and Marie-Francine Moens

16:10–16:35 *User Requirements Analysis for Meeting Information Retrieval Based on Query Elicitation*
Vincenzo Pallotta, Violeta Seretan and Marita Ailomaa

16:35–17:00 *Combining Multiple Knowledge Sources for Dialogue Segmentation in Multimedia Archives*
Pei-Yun Hsueh and Johanna D. Moore

Session 9D: Text mining and Retrieval

15:45–16:10 *Topic Analysis for Psychiatric Document Retrieval*
Liang-Chih Yu, Chung-Hsien Wu, Chin-Yew Lin, Eduard Hovy and Chia-Ling Lin

16:10–16:35 *What to be? - Electronic Career Guidance Based on Semantic Relatedness*
Iryna Gurevych, Christof Müller and Torsten Zesch

16:35–17:00 *Extracting Social Networks and Biographical Facts From Conversational Speech Transcripts*
Hongyan Jing, Nanda Kambhatla and Salim Roukos

17:10 Best Paper Award (Plenary Session)

Guiding Statistical Word Alignment Models With Prior Knowledge

Yonggang Deng and **Yuqing Gao**
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
{ydeng,yuqing}@us.ibm.com

Abstract

We present a general framework to incorporate prior knowledge such as heuristics or linguistic features in statistical generative word alignment models. Prior knowledge plays a role of probabilistic soft constraints between bilingual word pairs that shall be used to guide word alignment model training. We investigate knowledge that can be derived automatically from entropy principle and bilingual latent semantic analysis and show how they can be applied to improve translation performance.

1 Introduction

Statistical word alignment models learn word associations between parallel sentences from statistics. Most models are trained from corpora in an unsupervised manner whose success is heavily dependent on the quality and quantity of the training data. It has been shown that human knowledge, in the form of a small amount of manually annotated parallel data to be used to seed or guide model training, can significantly improve word alignment F-measure and translation performance (Ittycheriah and Roukos, 2005; Fraser and Marcu, 2006).

As formulated in the competitive linking algorithm (Melamed, 2000), the problem of word alignment can be regarded as a process of word linkage disambiguation, that is, choosing correct associations among all competing hypothesis. The more reasonable constraints are imposed on this process, the easier the task would become. For instance, the most relaxed IBM Model-1, which assumes that any source word can be generated by any target word equally regardless of distance, can be improved by demanding a Markov process of alignments as in HMM-based models (Vogel et al., 1996), or implementing a distribution of number of target words linked to a source word as in IBM fertility-based models (Brown et al., 1993).

Following the path, we shall put more constraints on word alignment models and investigate ways of implementing them in a statistical framework. We have seen examples showing that names tend to align to names and function words are likely to be linked to function words. These observations are independent of language and can be understood by common sense. Moreover, there are other linguistically motivated constraints. For instance, words aligned to each other presumably are semantically consistent; and likely to be, they are syntactically agreeable. In these paper, we shall exploit some of these constraints in building better word alignments in the application of statistical machine translation.

We propose a simple framework that can integrate prior knowledge into statistical word alignment model training. In the framework, prior knowledge serves as probabilistic soft constraints that will guide word alignment model training. We present two types of constraints that are derived in an unsupervised way: one is based on the entropy principle, the other comes from bilingual latent semantic analysis. We investigate their impact on word alignments and show their effectiveness in improving translation performance.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 1–8,
Prague, Czech Republic, June 2007.

2 Constrained Word Alignment Models

The framework that we propose to incorporate statistical constraints into word alignment models is generic. It can be applied to complicated models such IBM Model-4 (Brown et al., 1993). We shall take HMM-based word alignment model (Vogel et al., 1996) as an example and follow the notation of (Brown et al., 1993). Let $\mathbf{e} = e_1^l$ represent a source string and $\mathbf{f} = f_1^m$ a target string. The random variable $\mathbf{a} = a_1^m$ specifies the indices of source words that target words are aligned to.

In an HMM-based word alignment model, source words are treated as Markov states while target words are observations that are generated when jumping to states:

$$P(\mathbf{a}, \mathbf{f}|\mathbf{e}) = \prod_{j=1}^{m} P(a_j|a_{j-1}, \mathbf{e}) t(f_j|e_{a_j})$$

Notice that a target word f is generated from a source state e by a simple lookup of the translation table, a.k.a., t-table $t(f|e)$, as depicted in (A) of Figure 1. To incorporate prior knowledge or impose constraints, we introduce two nodes E and F representing the hidden tags of the source word e and the target word f respectively, and organize the dependency structure as in (B) of Figure 1. Given this generative procedure, f will also depend on its tag F, which is determined probabilistically by the source tag E. The dependency from E to F functions as a soft constraint showing how the two hidden tags are agreeable to each other. Mathematically, the conditional distribution follows:

$$\begin{aligned} P(f|e) &= \sum_{E,F} P(f, E, F|e) \\ &= \sum_{E,F} P(E|e) P(F|E) P(f|e, F) \\ &= t(f|e) \cdot Con(f, e), \end{aligned} \quad (1)$$

where

$$Con(f, e) = \sum_{E,F} P(E|e) P(F|E) P(F|f) / P(F) \quad (2)$$

is the soft weight attached to the t-table entry. It considers all possible hidden tags of e and f and serves as constraint between the link.

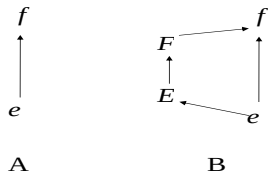


Figure 1: A simple table lookup (A) vs. a constrained procedure (B) of generating a target word f from a source word e.

We do not change the value of $Con(f, e)$ during iterative model training but rather keep it constant as an indicator of how strong the word pair should be considered as a candidate. This information is derived before word alignment model training and will act as soft constraints that need to be respected during training and alignments. For a given word pair, the soft constraint can have different assignment in different sentence pairs since the word tags can be context dependent.

To understand why we take the "detour" of generating a target word rather than directly from a t-table, consider the hidden tag as binary value indicating being a name or not. Without these constraints, t-table entries for names with low frequency tend to be flat and word alignments can be chosen randomly without sufficient statistics or strong lexical preference under maximum likelihood criterion. If we assume that a name is produced by a name with a high probability but by a non-name with a low probability, i.e. $P(F = E) >> P(F \neq E)$, proper names with low counts then are encouraged to link to proper names during training; and consequently, conditional probability mass would be more focused on correct name translations. On the other hand, names are discouraged to produce non-names. This will potentially avoid incorrect word associations. We are able to apply this type of constraint since usually there are many monolingual resources available to build a high performance probabilistic name tagger. The example suggests that putting reasonable constraints learned from monolingual analysis can alleviate data spareness problem in bilingual applications.

The weights $Con(f, e)$ are the prior knowledge that shall be assigned with care but respected during training. The baseline is to set all these weights

to 1, which is equivalent to placing no prior knowledge on model training. The introduction of these weights does not complicate parameter estimation procedure. Whenever a source word e is hypothesized to generate a target word f, the translation probability $t(f|e)$ should be weighted by $Con(f, e)$.

We point out that the constraints between f and e through their hidden tags are in probabilities. There are no hard decisions made before training. A strong preference between two words can be expressed by assigning corresponding weights close to 1. This will affect the final alignment model.

Depending on the hidden tags, there are many realizations of reasonable constraints that can be put beforehand. They can be semantic classes, syntactic annotations, or as simple as whether being a function word or content word. Moreover, the source side and the target side do not have to share the same set of tags. The framework is also flexible to support multiple types of constraints that can be implemented in parallel or cascaded sequence. Moreover, the constraints between words can be dependent on context within parallel sentences. Next, we will describe two types of constraints that we proposed. Both of them are derived from data in an unsupervised way.

2.1 Entropy Principle

It is assumed that generally speaking, a source function word generates a target function word with a higher probability than generating a target content word; similar assumption applies to a source content word as well. We capture this type of constraint by defining the hidden tag E and F as binary labels indicating being a content word or not. Based on the assumption, we design probabilistic relationship between the two hidden tags as:

$$P(E = F) = 1 - P(E \neq F) = \alpha,$$

where α is a scalar whose value is close to 1, say 0.9. The bigger α is, the tighter constraint we put on word pairs to be connected requiring the same type of label.

To determine the probability of a word being a function word, we apply the entropy principle. A function word, say "of","in" or "have", appears more frequently than a content word, say "journal" or "chemistry", in a document or sentence. We will approximate the probability of a word as a function word with the relative uncertainty of its being observed in a sentence.

More specifically, suppose we have N parallel sentences in the training corpus. For each word w_i[1], let c_{ij} be the number of word w_i observed in the j-th sentence pair, and let c_i be the total number of occurrences of w_i in the corpus. We define the relative entropy of word w_i as

$$\epsilon_{w_i} = -\frac{1}{\log N} \sum_{j=1}^{N} \frac{c_{ij}}{c_i} \log \frac{c_{ij}}{c_i}.$$

With the entropy of a word, the likelihood of word w being tagged as a function word is approximated with $w^{(1)} = \epsilon_w$ and being tagged as a content word with $w^{(0)} = 1 - \epsilon_w$.

We ignore the denominator in Equ. (2) and find the constraint under the entropy principle:

$$\begin{aligned} Con(f, e) &= \alpha(e^{(0)} f^{(0)} + e^{(1)} f^{(1)}) + \\ &\quad (1 - \alpha)(e^{(1)} f^{(0)} + e^{(0)} f^{(1)}). \end{aligned}$$

As can be seen, the connection between two words is simulated with a binary symmetric channel. An example distribution of the constraint function is illustrated in Figure 2. A high value of α encourages connecting word pairs with comparable entropy; When $\alpha = 0.5$, $Con(f, e)$ is constant which corresponds to applying no prior constraint; When α is close to 0, the function plays opposite role on word alignment training where a high frequency word is pushed to associate with a low frequency word.

2.2 Bilingual Latent Semantic Analysis

Latent Semantic Analysis (LSA) is a theory and method for extracting and representing the meaning of words by statistically analyzing word contextual usages in a collection of text. It provides a method by which to calculate the similarity of meaning of given words and documents. LSA has been successfully applied to information retrieval (Deerwester et al., 1990), statistical langauge modeling (Bellegarda, 2000) and etc.

[1]We prefix 'E_' to source words and 'F_' to target words to distinguish words that have the same spelling but are from different languages.

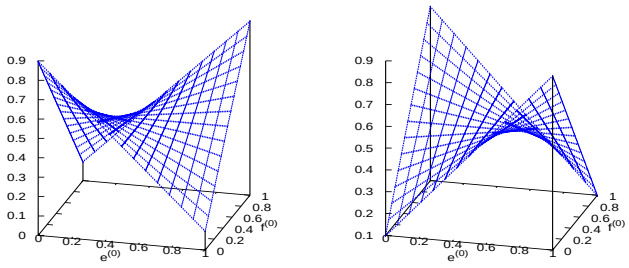


Figure 2: Distribution of the constraint function based on entropy principle when $\alpha = 0.9$ on the left and $\alpha = 0.1$ on the right.

We explore LSA techniques in bilingual environment to derive semantic constraints as prior knowledge for guiding a word alignment model training. The idea is to find semantic representation of source words and target words in the so-called low-dimensional LSA-space, and then to use their similarities to quantitatively establish semantic consistencies. We propose two different approaches.

2.2.1 A Simple Bag-of-word Model

One method we investigate is a simple bag-of-word model as in monolingual LSA. We treat each sentence pair as a document and do not distinguish source words and target words as if they are terms generated from the same vocabulary. A sparse matrix W characterizing word-document co-occurrence is constructed. Following the notation in section 2.1, the ij-th entry of the matrix W is defined as in (Bellegarda, 2000)

$$W_{ij} = (1 - \epsilon_{w_i}) \frac{c_{ij}}{c_j},$$

where c_j is the total number of words in the j-th sentence pair. This construction considers the importance of words globally (corpus wide) and locally (within sentence pairs). Alternative constructions of the matrix are possible using raw counts or TF-IDF (Deerwester et al., 1990).

W is a $M \times N$ sparse matrix, where M is the size of vocabulary including both source and target words. To obtain a compact representation, singular value decomposition (SVD) is employed (cf. Berry et al (1993)) to yield $W \approx \hat{W} = U \times S \times V^T$ as Figure 3 shows, where, for some order $R \ll \min(M, N)$ of the decomposition, U is a $M \times R$ left singular matrix with rows u_i, $i = 1, \cdots, M$, S is a $R \times R$ diagonal matrix of singular values $s_1 \geq s_2 \geq \ldots \geq s_R \gg 0$, and V is $N \times R$ a right singular matrix with rows v_j, $j = 1, \cdots, N$. For each i, the scaled R-vector $u_i S$ may be viewed as representing w_i, the i-th word in the vocabulary, and similarly the scaled R-vector $v_j S$ as representing d_j, j-th document in the corpus. Note that the $u_i S$'s and $v_j S$'s both belong to $\mathbb{R}^R$, the so-called LSA-space. All target and source words are projected into the same LSA-space too.

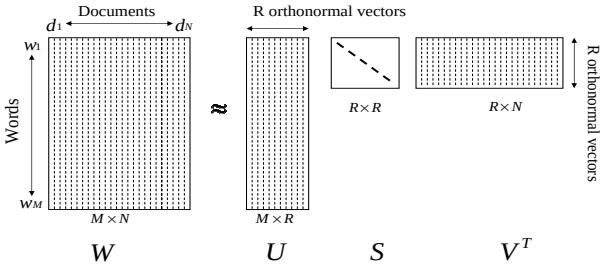


Figure 3: SVD of the Sparse Matrix W.

As Equ. (2) suggested, to induce semantic constraints in a straightforward way, one would proceed as follows: firstly, perform word semantic clustering with, say, their compact representations in the LSA-space; secondly, construct cluster generating dependencies by specifying the conditional distribution of $P(F|E)$; and finally, for each word pair, induce the semantic constraint by considering all possible semantic labeling schemes. We approximate this long process with simply finding word similarities defined by their cosine distance in the low dimension space:

$$Con(f, e) = \frac{1}{2}(\cos(u_f S, u_e S) + 1) \qquad (3)$$

The linear mapping above is introduced to avoid negative constraints and to set the maximum constraint value as 1.

In building word alignment models, a special "NULL" word is usually introduced to address target words that align to no source words. Since this physically non-existing word is not in the vocabulary of the bilingual LSA, we use the centroid of all source words as its vector representation in the LSA-space. The semantic constraints between "NULL" and any target words can be derived in the same way. However, this is chosen for mostly computational

convenience, and is not the only way to address the empty word issue.

2.2.2 Utilizing Word Alignment Statistics

While the simple bag-of-word model puts all source words and target words as rows in the matrix, another method of deriving semantic constraint constructs the sparse matrix by taking source words as rows and target words as columns and uses statistics from word alignment training to form word pair co-occurrence association.

More specifically, we regard each target word f as a "document" and each source word e as a "term". The number of occurrences of the source word e in the document f is defined as the expected number of times that f generates e in the parallel corpus under the word alignment model. This method requires training the baseline word alignment model in another direction by taking fs as source words and es as target words, which is often done for symmetric alignments, and then dumping out the soft counts when model converges. We threshold the minimum word-to-word translation probability to remove word pairs that have low co-occurrence counts.

Following the similarity induced semantic constraints in section 2.2.1, we need to find the distance between a term and a document. Let v_f be the projection of the document representing the target word f and u_e the projection of the term representing the source word e after performing SVD on the sparse matrix, we calculate the similarity between (f, e) and then find their semantic constraint to be

$$Con(f, e) = \frac{1}{2}(\cos(v_f S^{1/2}, u_e S^{1/2}) + 1) \qquad (4)$$

Unlike the method in section 2.2.1, there is no empty word issue here since we do have statistics of the "NULL" word as a source word generating e words and therefore there is a "document" assigned to it.

3 Experimental Results

We test our framework on the task of large vocabulary translation from dialectical (Iraqi) Arabic utterances into English. The task covers multiple domains including travel, emergency medical diagnosis, defense-oriented force protection, security and etc. To avoid impacts of speech recognition errors, we only report experiments from text to text translation.

The training corpus consists of 390K sentence pairs, with total 2.43M Arabic words and 3.38M English words. These sentences are in typical spoken transcription form, i.e., spelling errors, disfluencies, such as word or phrase repetition, and ungrammatical utterances are commonly observed. Arabic utterance length ranges from 3 to 70 words with the average of 6 words.

There are 25K entries in the English vocabulary and 90K in Arabic side. Data sparseness severely challenges word alignment model and consequently automatic phrase translation induction. There are 42K singletons in Arabic vocabulary, and 14K Arabic words with occurrence of twice each in the corpus. Since Arabic is a morphologically rich language where affixes are attached to stem words to indicate gender, tense, case and etc, in order to reduce vocabulary size and address out-of-vocabulary words, we split Arabic words into affix and root according to a rule-based segmentation scheme (Xiang et al., 2006) with the help from the Buckwalter analyzer (LDC, 2002) output. This reduces the size of Arabic vocabulary to 52K.

Our test data consists of 1294 sentence pairs. They are split into two parts: half of them is used as the development set, on which training parameters and decoding feature weights are tuned, the other half is for test.

3.1 Training and Translation Setup

Starting from the collection of parallel training sentences, we train word alignment models in two translation directions, from English to Iraqi Arabic and from Iraqi Arabic to English, and derive two sets of Viterbi alignments. By combining word alignments in two directions using heuristics (Och and Ney, 2003), a single set of static word alignments is then formed. All phrase pairs which respect to the word alignment boundary constraint are identified and pooled to build phrase translation tables with the Maximum Likelihood criterion. We prune phrase translation entries by their probabilities. The maximum number of tokens in Arabic phrases is set to 5 for all conditions.

Our decoder is a phrase-based multi-stack imple-

mentation of the log-linear model similar to Pharaoh (Koehn et al., 2003). Like other log-linear model based decoders, active features in our translation engine include translation models in two directions, lexicon weights in two directions, language model, distortion model, and sentence length penalty. These feature weights are tuned on the dev set to achieve optimal translation performance using downhill simplex method (Och and Ney, 2002). The language model is a statistical trigram model estimated with Modified Kneser-Ney smoothing (Chen and Goodman, 1996) using all English sentences in the parallel training data.

We measure translation performance by the BLEU score (Papineni et al., 2002) and Translation Error Rate (TER) (Snover et al., 2006) with one reference for each hypothesis. Word alignment models trained with different constraints are compared to show their effects on the resulting phrase translation tables and the final translation performance.

3.2 Translation Results

Our baseline word alignment model is the word-to-word Hidden Markov Model (Vogel et al., 1996). Basic models in two translation directions are trained simultaneously where statistics of two directions are shared to learn symmetric translation lexicon and word alignments with high precision motivated by (Zens et al., 2004) and (Liang et al., 2006). The baseline translation results (BLEU and TER) on the dev and test set are presented in the line "HMM" of Table 1. We also compare with results of IBM Model-4 word alignments implemented in GIZA++ toolkit (Och and Ney, 2003).

We study and compare two types of constraint and see how they affect word alignments and translation output. One is based on the entropy principle as described in Section 2.1, where α is set to 0.9; The other is based on bilingual latent semantic analysis.

For the simple bag-of-word bilingual LSA as described in Section 2.2.1, after SVD on the sparse matrix using the toolkit SVDPACK (Berry et al., 1993), all source and target words are projected into a low-dimensional ($R = 88$) LSA-space. Word pair semantic constrains are calculated based on their similarity as in Equ. 3 before word alignment training. Like the baseline, we perform 6 iterations of IBM Model-1 training and then 4 iteration of HMM training. The semantic constraints are used to guide word alignment model training for each iteration. The BLEU score and TER with this constraint are shown in the line "BiLSA-1" of Table 1.

To exploit word alignment statistics in bilingual LSA as described in Section 2.2.2, we dump out the statistics of the baseline word alignment model and use them to construct the sparse matrix. We find low-dimensional representation ($R = 67$) of English words and Arabic words and use their similarity to establish semantic constraints as in Equ. 4. The training procedure is the same as the baseline and "BiLSA-1". The translation results with these word alignments are shown as "BiLSA-2" in Table 1.

As Table 1 shows, when the entropy based constraints are applied, BLEU score improves 0.5 point on the test set. Clearly, when bilingual LSA constraints are applied, translation performance can be improved up to 1.6 BLEU points. We also observe that TER can drop 2.1 points with the "BiLSA-1" constraint.

While "BiLSA-1" constraint performs better on the test set, "BiLSA-2" constraint achieves slightly higher BLEU score on the dev set. We then try a simple combination of these two types of constraints, that is the geometric mean of $Con_{BiLSA-1}(f, e)$ and $Con_{BiLSA-2}(f, e)$, and find out that BLEU score can be improved a little bit further on both sets as the line "Mix" shows.

We notice that the relatively simpler HMM model can perform comparable or better than the sophisticated Model-4 when proper constraints are active in guiding word alignment model training. We also try to put constraints in Model-4. As the Equation 1 implies, when a word-to-word generative probability is needed, one should multiply corresponding lexicon entry in the t-table with the word pair constraint. We simply modify the GIZA++ toolkit (Och and Ney, 2003) by always weighting lexicon probabilities with soft constraints during iterative model training, and obtain 0.7% TER reduction on both sets and 0.4% BLEU improvement on the test set.

3.3 Analysis

To understand how prior knowledge encoded as soft constraints plays a role in guiding word alignment training, we compare statistics of different word alignment models. We find that our baseline HMM

Table 1: Translation Results with different word alignments.

Alignments	BLEU		TER	
	dev	test	dev	test
Model-4	0.310	0.296	0.528	0.530
+Mix	0.306	0.300	0.521	0.523
HMM	0.289	0.288	0.543	0.542
+Entropy	0.289	0.293	0.534	0.536
+BiLSA-1	0.294	0.300	0.531	0.521
+BiLSA-2	0.298	0.292	0.530	0.528
+Mix	0.302	0.304	0.532	0.524

generates 2.6% less number of total word links than that of Model-4. Part of the reason is that models of two directions in the baseline are trained simultaneously. The requirement of bi-directional evidence places a certain constraint on word alignments. When "BiLSA-1" constraints are applied in the baseline model, 2.7% less number of total word links are hypothesized, and consequently, less number of Arabic n-gram translations in the final phrase translation table are induced. The observation suggests that the constraints improve word alignment precision and accuracy of phrase translation tables as well.

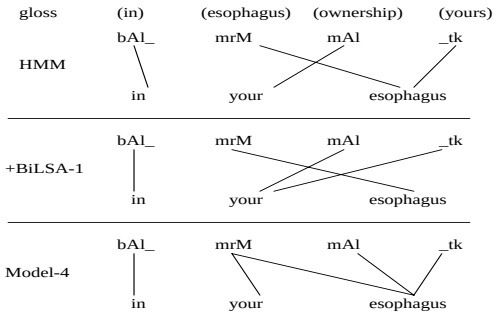


Figure 4: An example of word alignments under different models

Figure 4 shows example word alignments of a partial sentence pair. The complete English sentence is "have you ever had like any reflux diseases in your esophagus". We notice that the Arabic word "mrM" (means esophagus) appears only once in the corpus. Some of the word pair constraints are listed in Table 2. The example demos that due to reasonable constraints placed in word alignment training, the link to "_tK" is corrected and consequently we have accurate word translation for the Arabic singleton "mrM".

Table 2: Word pair constraint values

English e	Arabic f	$Con_{BiLSA-1}(f,e)$
esophagus	mrM	0.6424
	mAl	0.1819
	_tk	0.2897
your	mrM	0.6319
	mAl	0.4930
	_tk	0.9672

4 Related Work

Heuristics based on co-occurrence analysis, such as point-wise mutual information or Dice coefficients , have been shown to be indicative for word alignments (Zhang and Vogel, 2005; Melamed, 2000). The framework presented in this paper demonstrates the possibility of taking heuristics as constraints guiding statistical generative word alignment model training. Their effectiveness can be expected especially when data sparseness is severe.

Discriminative word alignment models, such as Ittycheriah and Roukos (2005); Moore (2005); Blunsom and Cohn (2006), have received great amount of study recently. They have proven that linguistic knowledge is useful in modeling word alignments under log-linear distributions as morphological, semantic or syntactic features. Our framework proposes to exploit these features differently by taking them as soft constraints of translation lexicon under a generative model.

While word alignments can help identifying semantic relations (van der Plas and Tiedemann, 2006), we proceed in the reverse direction. We investigate the impact of semantic constraints on statistical word alignment models as prior knowledge. In (Ma et al., 2004), bilingual semantic maps are constructed to guide word alignment. The framework we proposed seamlessly integrates derived semantic similarities into a statistical word alignment model. And we extended monolingual latent semantic analysis in bilingual applications.

Toutanova et al. (2002) augmented bilingual sentence pairs with part-of-speech tags as linguistic constraints for HMM-based word alignments. The constraints between tags are automatically learned in a parallel generative procedure along with lex-

icon. We have introduced hidden tags between a word pair to specialize their soft constraints, which serve as prior knowledge that will be used in guiding word alignment model training. Constraint between tags are embedded into the word to word generative process.

5 Conclusions and Future Work

We have presented a simple and effective framework to incorporate prior knowledge such as heuristics or linguistic features into statistical generative word alignment models. Prior knowledge serves as soft constraints that shall be placed on translation lexicon to guide word alignment model training and disambiguation during Viterbi alignment process. We studied two types of constraints that can be obtained automatically from data and showed improved performance (up to 1.6% absolute BLEU increase or 2.1% absolute TER reduction) in translating dialectical Arabic into English. Future work includes implementing the idea in alternative alignment models and also exploiting prior knowledge derived from such as manually-aligned data and pre-existing linguistic resources.

Acknowledgement We thank Mohamed Afify for discussions and the anonymous reviewers for suggestions.

References

J. R. Bellegarda. 2000. Exploiting latent semantic information in statistical language modeling. *Proc. of the IEEE*, 88(8):1279–1296, August.

M. Berry, T. Do, and S. Varadhan. 1993. Svdpackc (version 1.0) user's guide. Tech. report cs-93-194, University of Tennessee, Knoxville, TN.

P. Blunsom and T. Cohn. 2006. Discriminative word alignment with conditional random fields. In *Proc. of COLING/ACL*, pages 65–72.

P. Brown, S. Della Pietra, V. Della Pietra, and R. Mercer. 1993. The mathematics of machine translation: Parameter estimation. *Computational Linguistics*, 19:263–312.

S. F. Chen and J. Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proc. of ACL*, pages 310–318.

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.

A. Fraser and D. Marcu. 2006. Semi-supervised training for statistical word alignment. In *Proc. of COLING/ACL*, pages 769–776.

A. Ittycheriah and S. Roukos. 2005. A maximum entropy word aligner for arabic-english machine translation. In *Proc. of HLT/EMNLP*, pages 89–96.

P. Koehn, F. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proc. of HLT-NAACL*.

LDC, 2002. *Buckwalter Arabic Morphological Analyzer Version 1.0*. LDC Catalog Number LDC2002L49.

P. Liang, B. Taskar, and D. Klein. 2006. Alignment by agreement. In *Proc. of HLT/NAACL*, pages 104–111.

Q. Ma, K. Kanzaki, Y. Zhang, M. Murata, and H. Isahara. 2004. Self-organizing semantic maps and its application to word alignment in japanese-chinese parallel corpora. *Neural Netw.*, 17(8-9):1241–1253.

I. Dan. Melamed. 2000. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249.

R. C. Moore. 2005. A discriminative framework for bilingual word alignment. In *Proc. of HLT/EMNLP*, pages 81–88.

F. J. Och and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of ACL*, pages 295–302.

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

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL*, pages 311–318.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proc. of AMTA*.

K. Toutanova, H. T. Ilhan, and C. Manning. 2002. Extentions to HMM-based statistical word alignment models. In *Proc. of EMNLP*.

Lonneke van der Plas and Jörg Tiedemann. 2006. Finding synonyms using automatic word alignment and measures of distributional similarity. In *Proc. of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 866–873.

S. Vogel, H. Ney, and C. Tillmann. 1996. HMM based word alignment in statistical translation. In *Proc. of COLING*.

B. Xiang, K. Nguyen, L. Nguyen, R. Schwartz, and J. Makhoul. 2006. Morphological decomposition for arabic broadcast news transcription. In *Proc. of ICASSP*, pages 1089–1092.

R. Zens, E. Matusov, and H. Ney. 2004. Improved word alignment using a symmetric lexicon model. In *Proc. of COLING*, pages 36–42.

Y. Zhang and S. Vogel. 2005. Competitive grouping in integrated phrase segmentation and alignment model. In *Proc. of the ACL Workshop on Building and Using Parallel Texts*, pages 159–162.

A Discriminative Syntactic Word Order Model for Machine Translation

Pi-Chuan Chang*
Computer Science Department
Stanford University
Stanford, CA 94305
pichuan@stanford.edu

Kristina Toutanova
Microsoft Research
Redmond, WA
kristout@microsoft.com

Abstract

We present a global discriminative statistical word order model for machine translation. Our model combines syntactic movement and surface movement information, and is discriminatively trained to choose among possible word orders. We show that combining discriminative training with features to detect these two different kinds of movement phenomena leads to substantial improvements in word ordering performance over strong baselines. Integrating this word order model in a baseline MT system results in a 2.4 points improvement in BLEU for English to Japanese translation.

1 Introduction

The machine translation task can be viewed as consisting of two subtasks: predicting the collection of words in a translation, and deciding the order of the predicted words. For some language pairs, such as English and Japanese, the ordering problem is especially hard, because the target word order differs significantly from the source word order.

Previous work has shown that it is useful to model target language order in terms of movement of syntactic constituents in constituency trees (Yamada and Knight, 2001; Galley et al., 2006) or dependency trees (Quirk et al., 2005), which are obtained using a parser trained to determine linguistic constituency. Alternatively, order is modelled in terms of movement of automatically induced hierarchical structure of sentences (Chiang, 2005; Wu, 1997).

The advantages of modeling how a target language syntax tree moves with respect to a source language syntax tree are that (*i*) we can capture the fact that constituents move as a whole and generally respect the phrasal cohesion constraints (Fox, 2002), and (*ii*) we can model broad syntactic reordering phenomena, such as subject-verb-object constructions translating into subject-object-verb ones, as is generally the case for English and Japanese.

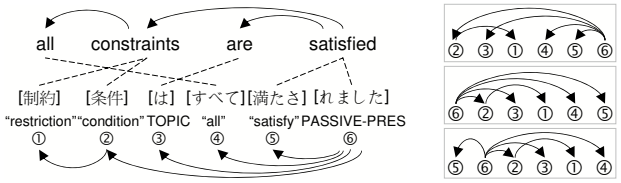
On the other hand, there is also significant amount of information in the surface strings of the source and target and their alignment. Many state-of-the-art SMT systems do not use trees and base the ordering decisions on surface phrases (Och and Ney, 2004; Al-Onaizan and Papineni, 2006; Kuhn et al., 2006). In this paper we develop an order model for machine translation which makes use of both syntactic and surface information.

The framework for our statistical model is as follows. We assume the existence of a dependency tree for the source sentence, an unordered dependency tree for the target sentence, and a word alignment between the target and source sentences. Figure 1 (a) shows an example of aligned source and target dependency trees. Our task is to order the target dependency tree.

We train a statistical model to select the best order of the unordered target dependency tree. An important advantage of our model is that it is global, and does not decompose the task of ordering a target sentence into a series of local decisions, as in the recently proposed order models for Machine Transition (Al-Onaizan and Papineni, 2006; Xiong et al., 2006; Kuhn et al., 2006). Thus we are able to define features over complete target sentence orders, and avoid the independence assumptions made by these

* This research was conducted during the author's internship at Microsoft Research.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 9–16,
Prague, Czech Republic, June 2007.



(a) (b)

Figure 1: (*a*) A sentence pair with source dependency tree, projected target dependency tree, and word alignments. (*b*) Example orders violating the target tree projectivity constraints.

models. Our model is discriminatively trained to select the best order (according to the BLEU measure) (Papineni et al., 2001) of an unordered target dependency tree from the space of possible orders.

Since the space of all possible orders of an unordered dependency tree is factorially large, we train our model on N-best lists of possible orders. These N-best lists are generated using approximate search and simpler models, as in the re-ranking approach of (Collins, 2000).

We first evaluate our model on the task of ordering target sentences, given correct (reference) unordered target dependency trees. Our results show that combining features derived from the source and target dependency trees, distortion surface order-based features (like the distortion used in Pharaoh (Koehn, 2004)) and language model-like features results in a model which significantly outperforms models using only some of the information sources.

We also evaluate the contribution of our model to the performance of an MT system. We integrate our order model in the MT system, by simply re-ordering the target translation sentences output by the system. The model resulted in an improvement from 33.6 to 35.4 BLEU points in English-to-Japanese translation on a computer domain.

2 Task Setup

The ordering problem in MT can be formulated as the task of ordering a target bag of words, given a source sentence and word alignments between target and source words. In this work we also assume a source dependency tree and an unordered target dependency tree are given. Figure 1(a) shows an example. We build a model that predicts an order of the target dependency tree, which induces an order on the target sentence words. The dependency tree constrains the possible orders of the target sentence only to the ones that are projective with respect to the tree. An order of the sentence is projective with respect to the tree if each word and its descendants form a contiguous subsequence in the ordered sentence. Figure 1(b) shows several orders of the sentence which violate this constraint.[1]

Previous studies have shown that if both the source and target dependency trees represent linguistic constituency, the alignment between subtrees in the two languages is very complex (Wellington et al., 2006). Thus such parallel trees would be difficult for MT systems to construct in translation. In this work only the source dependency trees are linguistically motivated and constructed by a parser trained to determine linguistic structure. The target dependency trees are obtained through projection of the source dependency trees, using the word alignment (we use GIZA++ (Och and Ney, 2004)), ensuring better parallelism of the source and target structures.

2.1 Obtaining Target Dependency Trees Through Projection

Our algorithm for obtaining target dependency trees by projection of the source trees via the word alignment is the one used in the MT system of (Quirk et al., 2005). We describe the algorithm schematically using the example in Figure 1. Projection of the dependency tree through alignments is not at all straightforward. One of the reasons of difficulty is that the alignment does not represent an isomorphism between the sentences, i.e. it is very often not a one-to-one and onto mapping.[2] If the alignment were one-to-one we could define the parent of a word w_t in the target to be the target word aligned to the parent of the source word s_i aligned to w_t. An additional difficulty is that such a definition could result in a non-projective target dependency tree. The projection algorithm of (Quirk et al., 2005) defines heuristics for each of these problems. In case of one-to-many alignments, for example, the case of "constraints" aligning to the Japanese words for "restriction" and "condition", the algorithm creates a

[1] For example, in the first order shown, the descendants of word 6 are not contiguous and thus this order violates the constraint.

[2] In an onto mapping, every word on the target side is associated with some word on the source side.

subtree in the target rooted at the rightmost of these words and attaches the other word(s) to it. In case of non-projectivity, the dependency tree is modified by re-attaching nodes higher up in the tree. Such a step is necessary for our example sentence, because the translations of the words "all" and "constraints" are not contiguous in the target even though they form a constituent in the source.

An important characteristic of the projection algorithm is that all of its heuristics use the *correct* target word order.[3] Thus the target dependency trees encode more information than is present in the source dependency trees and alignment.

2.2 Task Setup for Reference Sentences vs MT Output

Our model uses input of the same form when trained/tested on reference sentences and when used in machine translation: a source sentence with a dependency tree, an unordered target sentence with and unordered target dependency tree, and word alignments.

We train our model on reference sentences. In this setting, the given target dependency tree contains the correct bag of target words according to a reference translation, and is projective with respect to the correct word order of the reference by construction. We also evaluate our model in this setting; such an evaluation is useful because we can isolate the contribution of an order model, and develop it independently of an MT system.

When translating new sentences it is not possible to derive target dependency trees by the projection algorithm described above. In this setting, we use target dependency trees constructed by our baseline MT system (described in detail in 6.1). The system constructs dependency trees of the form shown in Figure 1 for each translation hypothesis. In this case the target dependency trees very often do not contain the correct target words and/or are not projective with respect to the best possible order.

[3]For example, checking which word is the rightmost for the heuristic for one-to-many mappings and checking whether the constructed tree is projective requires knowledge of the correct word order of the target.

3 Language Model with Syntactic Constraints: A Pilot Study

In this section we report the results of a pilot study to evaluate the difficulty of ordering a target sentence if we are given a target dependency tree as the one in Figure 1, versus if we are just given an unordered bag of target language words.

The difference between those two settings is that when ordering a target dependency tree, many of the orders of the sentence are not allowed, because they would be non-projective with respect to the tree. Figure 1 (b) shows some orders which violate the projectivity constraint. If the given target dependency tree is projective with respect to the correct word order, constraining the possible orders to the ones consistent with the tree can only help performance. In our experiments on reference sentences, the target dependency trees are projective by construction. If, however, the target dependency tree provided is not necessarily projective with respect to the best word order, the constraint may or may not be useful. This could happen in our experiments on ordering MT output sentences.

Thus in this section we aim to evaluate the usefulness of the constraint in both settings: reference sentences with projective dependency trees, and MT output sentences with possibly non-projective dependency trees. We also seek to establish a baseline for our task. Our methodology is to test a simple and effective order model, which is used by all state of the art SMT systems – a trigram language model – in the two settings: ordering an unordered bag of words, and ordering a target dependency tree.

Our experimental design is as follows. Given an unordered sentence t and an unordered target dependency tree $tree(t)$, we define two spaces of target sentence orders. These are the unconstrained space of all permutations, denoted by Permutations(t) and the space of all orders of t which are projective with respect to the target dependency tree, denoted by TargetProjective(t,$tree(t)$). For both spaces S, we apply a standard trigram target language model to select a most likely order from the space; i.e., we find a target order $order^*_S(t)$ such that: $order^*_S(t) = argmax_{order(t) \in S} Pr_{LM}(order(t))$. The operator which finds $order^*_S(t)$ is difficult to implement since the task is NP-hard in both set-

Reference Sentences		
Space	**BLEU**	**Avg. Size**
Permutations	58.8	2^{61}
TargetProjective	83.9	2^{29}
MT Output Sentences		
Space	**BLEU**	**Avg. Size**
Permutations	26.3	2^{56}
TargetProjective	31.7	2^{25}

Table 1: Performance of a tri-gram language model on ordering reference and MT output sentences: unconstrained or subject to target tree projectivity constraints.

tings, even for a bi-gram language model (Eisner and Tromble, 2006).[4] We implemented left-to-right beam A* search for the Permutations space, and a tree-based bottom up beam A* search for the TargetProjective space. To give an estimate of the search error in each case, we computed the number of times the correct order had a better language model score than the order returned by the search algorithm.[5] The lower bounds on search error were 4% for Permutations and 2% for TargetProjective, computed on reference sentences.

We compare the performance in BLEU of orders selected from both spaces. We evaluate the performance on reference sentences and on MT output sentences. Table 1 shows the results. In addition to BLEU scores, the table shows the median number of possible orders per sentence for the two spaces.

The highest achievable BLEU on reference sentences is 100, because we are given the correct bag of words. The highest achievable BLEU on MT output sentences is well below 100 (the BLEU score of the MT output sentences is 33). Table 3 describes the characteristics of the main data-sets used in the experiments in this paper; the test sets we use in the present pilot study are the reference test set (Ref-test) of 1K sentences and the MT test set (MT-test) of 1,000 sentences.

The results from our experiment show that the target tree projectivity constraint is extremely powerful on reference sentences, where the tree given is indeed projective. (Recall that in order to obtain the target dependency tree in this setting we have used information from the true order, which explains in part the large performance gain.)

[4]Even though the dependency tree constrains the space, the number of children of a node is not bounded by a constant.

[5]This is an underestimate of search error, because we don't know if there was another (non-reference) order which had a better score, but was not found.

The gain in BLEU due to the constraint was not as large on MT output sentences, but was still considerable. The reduction in search space size due to the constraint is enormous. There are about 2^{30} times fewer orders to consider in the space of target projective orders, compared to the space of all permutations. From these experiments we conclude that the constraints imposed by a projective target dependency tree are extremely informative. We also conclude that the constraints imposed by the target dependency trees constructed by our baseline MT system are very informative as well, even though the trees are not necessarily projective with respect to the best order. Thus the projectivity constraint with respect to a reasonably good target dependency tree is useful for addressing the search and modeling problems for MT ordering.

4 A Global Order Model for Target Dependency Trees

In the rest of the paper we present our new word order model and evaluate it on reference sentences and in machine translation. In line with previous work on NLP tasks such as parsing and recent work on machine translation, we develop a discriminative order model. An advantage of such a model is that we can easily combine different kinds of features (such as syntax-based and surface-based), and that we can optimize the parameters of our model directly for the evaluation measures of interest.

Additionally, we develop a globally normalized model, which avoids the independence assumptions in locally normalized conditional models.[6] We train a global log-linear model with a rich set of syntactic and surface features. Because the space of possible orders of an unordered dependency tree is factorially large, we use simpler models to generate N-best orders, which we then re-rank with a global model.

4.1 Generating N-best Orders

The simpler models which we use to generate N-best orders of the unordered target dependency trees are the standard trigram language model used in Section 3, and another statistical model, which we call a Local Tree Order Model (LTOM). The LTOM model

[6]Those models often assume that current decisions are independent of future observations.

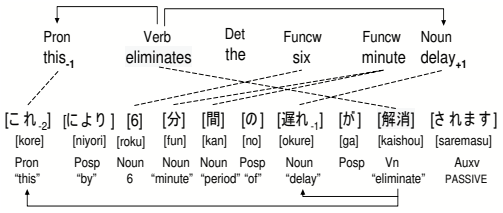


Figure 2: Dependency parse on the source (English) sentence, alignment and projected tree on the target (Japanese) sentence. Notice that the projected tree is only partial and is used to show the head-relative movement.

uses syntactic information from the source and target dependency trees, and orders each local tree of the target dependency tree independently. It follows the order model defined in (Quirk et al., 2005).

The model assigns a probability to the position of each target node (modifier) relative to its parent (head), based on information in both the source and target trees. The probability of an order of the complete target dependency tree decomposes into a product over probabilities of positions for each node in the tree as follows:

$$P(order(t)|s,t) = \prod_{n \in t} P(pos(n, parent(n))|s,t)$$

Here, position is modelled in terms of closeness to the head in the dependency tree. The closest pre-modifier of a given head has position -1; the closest post-modifier has a position 1. Figure 2 shows an example dependency tree pair annotated with head-relative positions. A small set of features is used to reflect local information in the dependency tree to model $P(pos(n, parent(n))|s,t)$: (*i*) lexical items of n and $parent(n)$, (*ii*) lexical items of the source nodes aligned to n and $parent(n)$, (*iii*) part-of-speech of the source nodes aligned to the node and its parent, and (*iv*) head-relative position of the source node aligned to the target node.

We train a log-linear model which uses these features on a training set of aligned sentences with source and target dependency trees in the form of Figure 2. The model is a local (non-sequence) classifier, because the decision on where to place each node does not depend on the placement of any other nodes.

Since the local tree order model learns to order whole subtrees of the target dependency tree, and since it uses syntactic information from the source, it provides an alternative view compared to the trigram language model. The example in Figure 2 shows that the head word "eliminates" takes a dependent "this" to the left (position -1), and on the Japanese side, the head word "kaishou" (corresponding to "eliminates") takes a dependent "kore" (corresponding to "this") to the left (position -2). The trigram language model would not capture the position of "kore" with respect to "kaishou", because the words are farther than three positions away.

We use the language model and the local tree order model to create N-best target dependency tree orders. In particular, we generate the N-best lists from a simple log-linear combination of the two models:

$$P(o(t)|s,t) \propto P_{LM}(o(t)|t) P_{LTOM}(o(t)|s,t)^{\lambda}$$

where $o(t)$ denotes an order of the target.[7] We used a bottom-up beam A* search to generate N-best orders. The performance of each of these two models and their combination, together with the 30-best oracle performance on reference sentences is shown in Table 2. As we can see, the 30-best oracle performance of the combined model (98.0) is much higher than the 1-best performance (92.6) and thus there is a lot of room for improvement.

4.2 Model

The log-linear reranking model is defined as follows. For each sentence pair sp_l ($l = 1, 2, ..., L$) in the training data, we have N candidate target word orders $o_{l,1}, o_{l,2}, ..., o_{l,N}$, which are the orders generated from the simpler models. Without loss of generality, we define $o_{l,1}$ to be the order with the highest BLEU score with respect to the correct order.[8]

We define a set of feature functions $f_m(o_{l,n}, sp_l)$ to describe a target word order $o_{l,n}$ of a given sentence pair sp_l. In the log-linear model, a corresponding weights vector λ is used to define the distribution over all possible candidate orders:

$$p(o_{l,n}|sp_l, \lambda) = \frac{e^{\lambda F(o_{l,n}, sp_l)}}{\sum_{n'} e^{\lambda F(o_{l,n'}, sp_l)}}$$

[7] We used the value $\lambda = .5$, which we selected on a development set to maximize BLEU.

[8] To avoid the problem that all orders could have a BLEU score of 0 if none of them contains a correct word four-gram, we define sentence-level k-gram BLEU, where k is the highest order, $k \leq 4$, for which there exists a correct k-gram in at least one of the N-Best orders.

We train the parameters λ by minimizing the negative log-likelihood of the training data plus a quadratic regularization term:

$L(\lambda) = -\sum_l \log p(o_{l,1}|sp_i, \lambda) + \frac{1}{2\sigma^2}\sum_m {\lambda_m}^2$

We also explored maximizing expected BLEU as our objective function, but since it is not convex, the performance was less stable and ultimately slightly worse, as compared to the log-likelihood objective.

4.3 Features

We design features to capture both the head-relative movement and the surface sequence movement of words in a sentence. We experiment with different combinations of features and show their contribution in Table 2 for reference sentences and Table 4 in machine translation. The notations used in the tables are defined as follows:

Baseline: **LTOM+LM** as described in Section 4.1

Word Bigram: Word bigrams of the target sentence. Examples from Figure 2: *"kore"+"niyori"*, *"niyori"+"roku"*.

DISP: Displacement feature. For each word position in the target sentence, we examine the alignment of the current word and the previous word, and categorize the possible patterns into 3 kinds: (a) parallel, (b) crossing, and (c) widening. Figure 3 shows how these three categories are defined.

Pharaoh DISP: Displacement as used in Pharaoh (Koehn, 2004). For each position in the sentence, the value of the feature is one less than the difference (absolute value) of the positions of the source words aligned to the current and the previous target word.

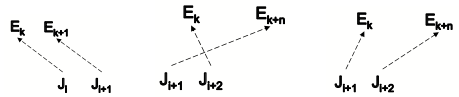
POSs and **POSt**: POS tags on the source and target sides. For Japanese, we have a set of 19 POS tags.

'+' means making conjunction of features and $prev()$ means using the information associated with the word from position -1.

In all explored models, we include the log-probability of an order according to the language model and the log-probability according to the local tree order model, the two features used by the baseline model.

5 Evaluation on Reference Sentences

Our experiments on ordering reference sentences use a set of 445K English sentences with their reference Japanese translations. This is a subset of the



(a) parallel (b) crossing (c) widening

Figure 3: Displacement feature: different alignment patterns of two contiguous words in the target sentence.

set MT-train in Table 3. The sentences were annotated with alignment (using GIZA++ (Och and Ney, 2004)) and syntactic dependency structures of the source and target, obtained as described in Section 2. Japanese POS tags were assigned by an automatic POS tagger, which is a local classifier not using tag sequence information.

We used 400K sentence pairs from the complete set to train the first pass models: the language model was trained on 400K sentences, and the local tree order model was trained on 100K of them. We generated N-best target tree orders for the rest of the data (45K sentence pairs), and used it for training and evaluating the re-ranking model. The re-ranking model was trained on 44K sentence pairs. All models were evaluated on the remaining 1,000 sentence pairs set, which is the set Ref-test in Table 3.

The top part of Table 2 presents the 1-best BLEU scores (actual performance) and 30-best oracle BLEU scores of the first-pass models and their log-linear combination, described in Section 4. We can see that the combination of the language model and the local tree order model outperformed either model by a large margin. This indicates that combining syntactic (from the LTOM model) and surface-based (from the language model) information is very effective even at this stage of selecting N-best orders for re-ranking. According to the 30-best oracle performance of the combined model LTOM+LM, 98.0 BLEU is the upper bound on performance of our re-ranking approach.

The bottom part of the table shows the performance of the global log-linear model, when features in addition to the scores from the two first-pass models are added to the model. Adding word-bigram features increased performance by about 0.6 BLEU points, indicating that training language-model like features discriminatively to optimize ordering performance, is indeed worthwhile. Next we compare

First-pass models		
Model	BLEU	
	1 best	30 best
Lang Model (Permutations)	58.8	71.2
Lang Model (TargetProjective)	83.9	95.0
Local Tree Order Model	75.8	87.3
Local Tree Order Model + Lang Model	92.6	98.0
Re-ranking Models		
Features	BLEU	
Baseline	92.60	
Word Bigram	93.19	
Pharaoh DISP	92.94	
DISP	93.57	
DISP+POSs	94.04	
DISP+POSs+POSt	94.14	
DISP+POSs+POSt, prev(DISP)+POSs+POSt	94.34	
DISP+POSs+POSt, prev(DISP)+POSs+POSt, WB	94.50	

Table 2: Performance of the first-pass order models and 30-best oracle performance, followed by performance of re-ranking model for different feature sets. Results are on reference sentences.

the Pharaoh displacement feature to the displacement feature we illustrated in Figure 3. We can see that the Pharaoh displacement feature improves performance of the baseline by .34 points, whereas our displacement feature improves performance by nearly 1 BLEU point. Concatenating the DISP feature with the POS tag of the source word aligned to the current word improved performance slightly.

The results show that surface movement features (i.e. the DISP feature) improve the performance of a model using syntactic-movement features (i.e. the LTOM model). Additionally, adding part-of-speech information from both languages in combination with displacement, and using a higher order on the displacement features was useful. The performance of our best model, which included all information sources, is 94.5 BLEU points, which is a 35% improvement over the fist-pass models, relative to the upper bound.

6 Evaluation in Machine Translation

We apply our model to machine translation by reordering the translation produced by a baseline MT system. Our baseline MT system constructs, for each target translation hypothesis, a target dependency tree. Thus we can apply our model to MT output in exactly the same way as for reference sentences, but using much noisier input: a source sentence with a dependency tree, word alignment and an unordered target dependency tree as the example shown in Figure 2. The difference is that the target dependency tree will likely not contain the correct target words and/or will not be projective with respect to the best possible order.

data set	num sent.	English		Japanese	
		avg. len	vocab	avg. len	vocab
MT-train	500K	15.8	77K	18.7	79K
MT-test	1K	17.5	–	20.9	–
Ref-test	1K	17.5	–	21.2	–

Table 3: Main data sets used in experiments.

6.1 Baseline MT System

Our baseline SMT system is the system of Quirk et al. (2005). It translates by first deriving a dependency tree for the source sentence and then translating the source dependency tree to a target dependency tree, using a set of probabilistic models. The translation is based on treelet pairs. A treelet is a connected subgraph of the source or target dependency tree. A treelet translation pair is a pair of word-aligned source and target treelets.

The baseline SMT model combines this treelet translation model with other feature functions — a target language model, a tree order model, lexical weighting features to smooth the translation probabilities, word count feature, and treelet-pairs count feature. These models are combined as feature functions in a (log)linear model for predicting a target sentence given a source sentence, in the framework proposed by (Och and Ney, 2002). The weights of this model are trained to maximize BLEU (Och and Ney, 2004). The SMT system is trained using the same form of data as our order model: parallel source and target dependency trees as in Figure 2.

Of particular interest are the components in the baseline SMT system contributing most to word order decisions. The SMT system uses the same target language trigram model and local tree order model, as we are using for generating N-best orders for reranking. Thus the baseline system already uses our first-pass order models and only lacks the additional information provided by our re-ranking order model.

6.2 Data and Experimental Results

The baseline MT system was trained on the MT-train dataset described in Table 3. The test set for the MT experiment is a 1K sentences set from the same domain (shown as MT-test in the table). The weights in the linear model used by the baseline SMT system were tuned on a separate development set.

Table 4 shows the performance of the first-pass models in the top part, and the performance of our

First-pass models		
Model	BLEU	
	1 best	30 best
Baseline MT System	33.0	–
Lang Model (Permutations)	26.3	28.7
Lang Model (TargetCohesive)	31.7	35.0
Local Tree Order Model	27.2	31.5
Local Tree Order Model + Lang Model	33.6	36.0
Re-ranking Models		
Features	BLEU	
Baseline	33.56	
Word Bigram	34.11	
Pharaoh DISP	34.67	
DISP	34.90	
DISP+POSs	35.28	
DISP+POSs+POSt	35.22	
DISP+POSs+POSt, prev(DISP)+POSs+POSt	35.33	
DISP+POSs+POSt, prev(DISP)+POSs+POSt, WB	35.37	

Table 4: Performance of the first pass order models and 30-best oracle performance, followed by performance of re-ranking model for different feature sets. Results are in MT.

re-ranking model in the bottom part. The first row of the table shows the performance of the baseline MT system, which is a BLEU score of 33. Our first-pass and re-ranking models re-order the words of this 1-best output from the MT system. As for reference sentences, the combination of the two first-pass models outperforms the individual models. The 1-best performance of the combination is 33.6 and the 30-best oracle is 36.0. Thus the best we could do with our re-ranking model in this setting is 36 BLEU points.[9] Our best re-ranking model achieves 2.4 BLEU points improvement over the baseline MT system and 1.8 points improvement over the first-pass models, as shown in the table. The trends here are similar to the ones observed in our reference experiments, with the difference that target POS tags were less useful (perhaps due to ungrammatical candidates) and the displacement features were more useful. We can see that our re-ranking model almost reached the upper bound oracle performance, reducing the gap between the first-pass models performance (33.6) and the oracle (36.0) by 75%.

7 Conclusions and Future Work

We have presented a discriminative syntax-based order model for machine translation, trained to to select from the space of orders projective with respect to a target dependency tree. We investigated a combination of features modeling surface movement and syntactic movement phenomena and showed that these two information sources are complementary and their combination is powerful. Our results on ordering MT output and reference sentences were very encouraging. We obtained substantial improvement by the simple method of post-processing the 1-best MT output to re-order the proposed translation. In the future, we would like to explore tighter integration of our order model with the SMT system and to develop more accurate algorithms for constructing projective target dependency trees in translation.

[9] Notice that the combination of our two first-pass models outperforms the baseline MT system by half a point (33.6 versus 33.0). This is perhaps due to the fact that the MT system searches through a much larger space (possible word translations in addition to word orders), and thus could have a higher search error.

References

Y. Al-Onaizan and K. Papineni. 2006. Distortion models for statistical machine translation. In *ACL*.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *ACL*.

M. Collins. 2000. Discriminative reranking for natural language parsing. In *ICML*, pages 175–182.

J Eisner and R. W. Tromble. 2006. Local search with very large-scale neighborhoods for optimal permutations in machine translation. In *HLT-NAACL Workshop*.

H. Fox. 2002. Phrasal cohesion and statistical machine translation. In *EMNLP*.

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 *ACL*.

P. Koehn. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *AMTA*.

R. Kuhn, D. Yuen, M. Simard, P. Paul, G. Foster, E. Joanis, and H. Johnson. 2006. Segment choice models: Feature-rich models for global distortion in statistical machine translation. In *HLT-NAACL*.

F. J. Och and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *ACL*.

F. J. Och and H. Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4).

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. In *ACL*.

C. Quirk, A. Menezes, and C. Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *ACL*.

B. Wellington, S. Waxmonsky, and I. Dan Melamed. 2006. Empirical lower bounds on the complexity of translational equivalence. In *ACL-COLING*.

D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

D. Xiong, Q. Liu, and S. Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *ACL*.

K. Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *ACL*.

Tailoring Word Alignments to Syntactic Machine Translation

John DeNero
Computer Science Division
University of California, Berkeley
denero@berkeley.edu

Dan Klein
Computer Science Division
University of California, Berkeley
klein@cs.berkeley.edu

Abstract

Extracting tree transducer rules for syntactic MT systems can be hindered by word alignment errors that violate syntactic correspondences. We propose a novel model for unsupervised word alignment which explicitly takes into account target language constituent structure, while retaining the robustness and efficiency of the HMM alignment model. Our model's predictions improve the yield of a tree transducer extraction system, without sacrificing alignment quality. We also discuss the impact of various posterior-based methods of reconciling bidirectional alignments.

1 Introduction

Syntactic methods are an increasingly promising approach to statistical machine translation, being both algorithmically appealing (Melamed, 2004; Wu, 1997) and empirically successful (Chiang, 2005; Galley et al., 2006). However, despite recent progress, almost all syntactic MT systems, indeed statistical MT systems in general, build upon crude legacy models of word alignment. This dependence runs deep; for example, Galley et al. (2006) requires word alignments to project trees from the target language to the source, while Chiang (2005) requires alignments to induce grammar rules.

Word alignment models have not stood still in recent years. Unsupervised methods have seen substantial reductions in alignment error (Liang et al., 2006) as measured by the now much-maligned AER metric. A host of discriminative methods have been introduced (Taskar et al., 2005; Moore, 2005; Ayan and Dorr, 2006). However, few of these methods have explicitly addressed the tension between word alignments and the syntactic processes that employ them (Cherry and Lin, 2006; Daumé III and Marcu, 2005; Lopez and Resnik, 2005).

We are particularly motivated by systems like the one described in Galley et al. (2006), which constructs translations using tree-to-string transducer rules. These rules are extracted from a bitext annotated with both English (target side) parses and word alignments. Rules are extracted from target side constituents that can be projected onto contiguous spans of the source sentence via the word alignment. Constituents that project onto non-contiguous spans of the source sentence do not yield transducer rules themselves, and can only be incorporated into larger transducer rules. Thus, if the word alignment of a sentence pair does not respect the constituent structure of the target sentence, then the minimal translation units must span large tree fragments, which do not generalize well.

We present and evaluate an unsupervised word alignment model similar in character and computation to the HMM model (Ney and Vogel, 1996), but which incorporates a novel, syntax-aware distortion component which conditions on target language parse trees. These trees, while automatically generated and therefore imperfect, are nonetheless (1) a useful source of structural bias and (2) the same trees which constrain future stages of processing anyway. In our model, the trees do not *rule out* any alignments, but rather softly influence the probability of transitioning between alignment positions. In particular, transition probabilities condition upon paths through the target parse tree, allowing the model to prefer distortions which respect the tree structure.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 17–24,
Prague, Czech Republic, June 2007.

Our model generates word alignments that better respect the parse trees upon which they are conditioned, without sacrificing alignment quality. Using the joint training technique of Liang et al. (2006) to initialize the model parameters, we achieve an AER superior to the GIZA++ implementation of IBM model 4 (Och and Ney, 2003) and a reduction of 56.3% in aligned interior nodes, a measure of agreement between alignments and parses. As a result, our alignments yield more rules, which better match those we would extract had we used manual alignments.

2 Translation with Tree Transducers

In a tree transducer system, as in phrase-based systems, the coverage and generality of the transducer inventory is strongly related to the effectiveness of the translation model (Galley et al., 2006). We will demonstrate that this coverage, in turn, is related to the degree to which initial word alignments respect syntactic correspondences.

2.1 Rule Extraction

Galley et al. (2004) proposes a method for extracting tree transducer rules from a parallel corpus. Given a source language sentence s, a target language parse tree t of its translation, and a word-level alignment, their algorithm identifies the constituents in t which map onto contiguous substrings of s via the alignment. The root nodes of such constituents – denoted *frontier nodes* – serve as the roots and leaves of tree fragments that form *minimal* transducer rules.

Frontier nodes are distinguished by their compatibility with the word alignment. For a constituent c of t, we consider the set of source words s_c that are aligned to c. If none of the source words in the linear closure s_c^* (the words between the leftmost and rightmost members of s_c) aligns to a target word outside of c, then the root of c is a frontier node. The remaining *interior* nodes do not generate rules, but can play a secondary role in a translation system.[1] The roots of null-aligned constituents are not frontier nodes, but can attach productively to multiple minimal rules.

[1]Interior nodes can be used, for instance, in evaluating syntax-based language models. They also serve to differentiate transducer rules that have the same frontier nodes but different internal structure.

Two transducer rules, $t_1 \rightarrow s_1$ and $t_2 \rightarrow s_2$, can be combined to form larger translation units by composing t_1 and t_2 at a shared frontier node and appropriately concatenating s_1 and s_2. However, no technique has yet been shown to robustly extract smaller component rules from a large transducer rule. Thus, for the purpose of maximizing the coverage of the extracted translation model, we prefer to extract many small, minimal rules and generate larger rules via composition. Maximizing the number of frontier nodes supports this goal, while inducing many aligned interior nodes hinders it.

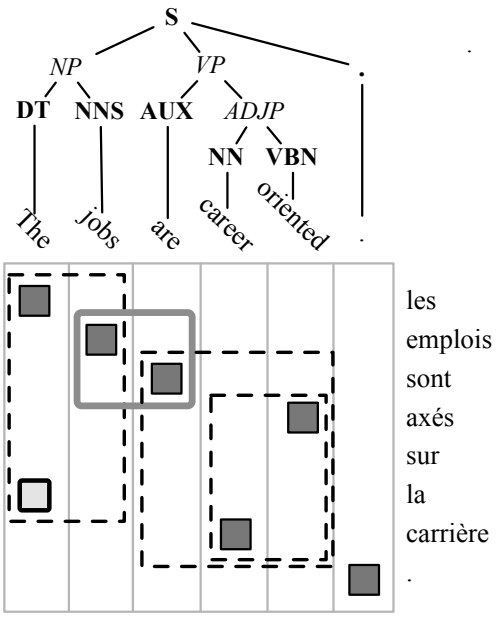
2.2 Word Alignment Interactions

We now turn to the interaction between word alignments and the transducer extraction algorithm. Consider the example sentence in figure 1A, which demonstrates how a particular type of alignment error prevents the extraction of many useful transducer rules. The mistaken link [*la* ⇒ *the*] intervenes between *axés* and *carrièr*, which both align within an English adjective phrase, while *la* aligns to a distant subspan of the English parse tree. In this way, the alignment violates the constituent structure of the English parse.

While alignment errors are undesirable in general, this error is particularly problematic for a syntax-based translation system. In a phrase-based system, this link would block extraction of the phrases [*axés sur la carrièr* ⇒ *career oriented*] and [*les emplois* ⇒ *the jobs*] because the error overlaps with both. However, the intervening phrase [*emplois sont* ⇒ *jobs are*] would still be extracted, at least capturing the transfer of subject-verb agreement. By contrast, the tree transducer extraction method fails to extract any of these fragments: the alignment error causes all non-terminal nodes in the parse tree to be interior nodes, excluding preterminals and the root. Figure 1B exposes the consequences: a wide array of desired rules are lost during extraction.

The degree to which a word alignment respects the constituent structure of a parse tree can be quantified by the frequency of interior nodes, which indicate alignment patterns that cross constituent boundaries. To achieve maximum coverage of the translation model, we hope to infer tree-violating alignments only when syntactic structures truly diverge.

(A)



Legend

Bold	Frontier node (agrees with alignment)
Italic	Interior node (inconsistent with alignment)

■	*Correct* proposed word alignment consistent with human annotation.
■	Proposed word alignment *error* inconsistent with human annotation.
▭	Word alignment constellation that renders the root of the relevant constituent to be an interior node.
▭	Word alignment constellation that would allow a phrase extraction in a phrase-based translation system, but which does not correspond to an English constituent.

(B) (i)

(S NP[0] VP[1] .[2]) → [0] [1] [2]
(S (NP (DT[0] NNS[1]) VP[2] .[3]) → [0] [1] [2] [3]
(S (NP (DT[0] (NNS *jobs*)) VP[2] .[3]) → [0] *emplois* [2] [3]
(S (NP (DT[0] (NNS *jobs*)) (VP AUX[1] ADJV[2]) .[3]) → [0] *emplois* [1] [2] [3]
(S (NP (DT[0] (NNS *jobs*)) (VP (AUX *are*) ADJV[1]) .[2]) → [0] *emplois sont* [1] [2]

(ii)

(S (NP (DT[0] NNS[1]) (VP AUX[2] (ADJV NN[3] VBN[4]) .[5]) → [0] [1] [2] [3] [4] [5]
(S (NP (DT[0] (NNS *jobs*)) (VP AUX[1] (ADJV NN[2] VBN[3]) .[4]) → [0] *sont* [1] [2] [3] [4]
(S (NP (DT[0] (NNS *jobs*)) (VP (AUX *are*) (ADJV NN[1] VBN[2]) .[3]) → [0] *emplois sont* [1] [2] [3]

Figure 1: In this transducer extraction example, (A) shows a proposed alignment from our test set with an alignment error that violates the constituent structure of the English sentence. The resulting frontier nodes are printed in bold; all nodes would be frontier nodes under a correct alignment. (B) shows a small sample of the rules extracted under the proposed alignment, (ii), and the correct alignment, (i) and (ii). The single alignment error prevents the extraction of all rules in (i) and many more. This alignment pattern was observed in our test set and corrected by our model.

3 Unsupervised Word Alignment

To allow for this preference, we present a novel conditional alignment model of a foreign (source) sentence $\mathbf{f} = \{f_1, ..., f_J\}$ given an English (target) sentence $\mathbf{e} = \{e_1, ..., e_I\}$ and a target tree structure t. Like the classic IBM models (Brown et al., 1994), our model will introduce a latent alignment vector $\mathbf{a} = \{a_1, ..., a_J\}$ that specifies the position of an aligned target word for each source word. Formally, our model describes $p(\mathbf{a}, \mathbf{f}|\mathbf{e}, t)$, but otherwise borrows heavily from the HMM alignment model of Ney and Vogel (1996).

The HMM model captures the intuition that the alignment vector $\mathbf{a}$ will in general progress across the sentence $\mathbf{e}$ in a pattern which is mostly local, perhaps with a few large jumps. That is, alignments are locally monotonic more often than not.

Formally, the HMM model factors as:

$$p(\mathbf{a}, \mathbf{f}|\mathbf{e}) = \prod_{j=1}^{J} p_d(a_j|a_{j_-}, j) p_\ell(f_j|e_{a_j})$$

where j_- is the position of the last non-null-aligned source word before position j, p_ℓ is a lexical transfer model, and p_d is a local distortion model. As in all such models, the lexical component p_ℓ is a collection of unsmoothed multinomial distributions over

foreign words.

The distortion model $p_d(a_j|a_{j_-}, j)$ is a distribution over the signed distance $a_j - a_{j_-}$, typically parameterized as a multinomial, Gaussian or exponential distribution. The implementation that serves as our baseline uses a multinomial distribution with separate parameters for $j = 1$, $j = J$ and shared parameters for all $1 < j < J$. Null alignments have fixed probability at any position. Inference over **a** requires only the standard forward-backward algorithm.

3.1 Syntax-Sensitive Distortion

The broad and robust success of the HMM alignment model underscores the utility of its assumptions: that word-level translations can be usefully modeled via first-degree Markov transitions and independent lexical productions. However, its distortion model considers only string distance, disregarding the constituent structure of the English sentence.

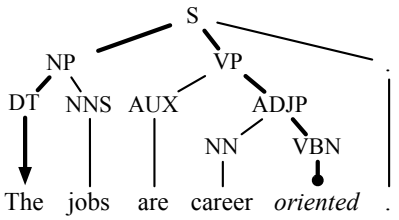
To allow syntax-sensitive distortion, we consider a new distortion model of the form $p_d(a_j|a_{j_-}, j, t)$. We condition on t via a generative process that transitions between two English positions by traversing the unique shortest path $\rho_{(a_{j_-}, a_j, t)}$ through t from a_{j_-} to a_j. We constrain ourselves to this shortest path using a staged generative process.

Stage 1 (POP$(\hat{n})$, STOP$(\hat{n})$): Starting in the leaf node at a_{j_-}, we choose whether to STOP or POP from child to parent, conditioning on the type of the parent node $\hat{n}$. Upon choosing STOP, we transition to stage 2.

Stage 2 (MOVE$(\hat{n}, d)$): Again, conditioning on the type of the parent $\hat{n}$ of the current node n, we choose a sibling $\bar{n}$ based on the signed distance $d = \phi_{\hat{n}}(n) - \phi_{\hat{n}}(\bar{n})$, where $\phi_{\hat{n}}(n)$ is the index of n in the child list of $\hat{n}$. Zero distance moves are disallowed. After exactly one MOVE, we transition to stage 3.

Stage 3 (PUSH$(n, \phi_n(\breve{n}))$): Given the current node n, we select one of its children $\breve{n}$, conditioning on the type of n and the position of the child $\phi_n(\breve{n})$. We continue to PUSH until reaching a leaf.

This process is a first-degree Markov walk through the tree, conditioning on the current node and its immediate surroundings at each step. We enforce the property that $\rho_{(a_{j_-}, a_j, t)}$ be unique by staging the process and disallowing zero distance moves in stage 2. Figure 2 gives an example sequence of tree transitions for a small parse tree.



Stage 1: { Pop(VBN), Pop(ADJP), Pop(VP), Stop(S) }
Stage 2: { Move(S, -1) }
Stage 3: { Push(NP, 1), Push(DT, 1) }

Figure 2: An example sequence of staged tree transitions implied by the unique shortest path from the word *oriented* ($a_{j_-} = 5$) to the word *the* ($a_j = 1$).

The parameterization of this distortion model follows directly from its generative process. Given a path $\rho_{(a_{j_-}, a_j, t)}$ with $r = k + m + 3$ nodes including the two leaves, the nearest common ancestor, k intervening nodes on the ascent and m on the descent, we express it as a triple of staged tree transitions that include k POPs, a STOP, a MOVE, and m PUSHes:

$$\begin{pmatrix} \{\text{POP}(n_2), ..., \text{POP}(n_{k+1}), \text{STOP}(n_{k+2})\} \\ \{\text{MOVE}\,(n_{k+2}, \phi(n_{k+3}) - \phi(n_{k+1}))\} \\ \{\text{PUSH}\,(n_{k+3}, \phi(n_{k+4}))\,, ..., \text{PUSH}\,(n_{r-1}, \phi(n_r))\} \end{pmatrix}$$

Next, we assign probabilities to each tree transition in each stage. In selecting these distributions, we aim to maintain the original HMM's sensitivity to target word order:

- Selecting POP or STOP is a simple Bernoulli distribution conditioned upon a node type.
- We model both MOVE and PUSH as multinomial distributions over the signed distance in positions (assuming a starting position of 0 for PUSH), echoing the parameterization popular in implementations of the HMM model.

This model reduces to the classic HMM distortion model given minimal English trees of only uniformly labeled pre-terminals and a root node. The classic 0-distance distortion would correspond to the

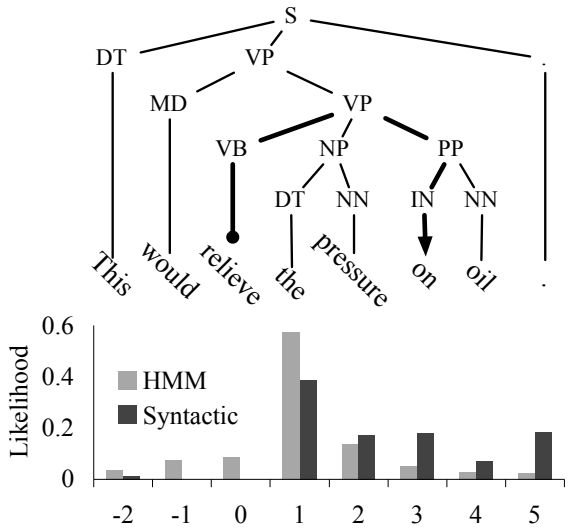


Figure 3: For this example sentence, the learned distortion distribution of $p_d(a_j|a_{j_-}, j, t)$ resembles its counterpart $p_d(a_j|a_{j_-}, j)$ of the HMM model but reflects the constituent structure of the English tree t. For instance, the short path from *relieve* to *on* gives a high transition likelihood.

STOP probability of the pre-terminal label; all other distances would correspond to MOVE probabilities conditioned on the root label, and the probability of transitioning to the terminal state would correspond to the POP probability of the root label.

As in a multinomial-distortion implementation of the classic HMM model, we must sometimes artificially normalize these distributions in the deficient case that certain jumps extend beyond the ends of the local rules. For this reason, MOVE and PUSH are actually parameterized by three values: a node type, a signed distance, and a range of options that dictates a normalization adjustment.

Once each tree transition generates a score, their product gives the probability of the entire path, and thereby the cost of the transition between string positions. Figure 3 shows an example learned distribution that reflects the structure of the given parse.

With these derivation steps in place, we must address a handful of special cases to complete the generative model. We require that the Markov walk from leaf to leaf of the English tree must start and end at the root, using the following assumptions.

1. Given no previous alignment, we forego stages 1 and 2 and begin with a series of PUSHes from the root of the tree to the desired leaf.

2. Given no subsequent alignments, we skip stages 2 and 3 after a series of POPs including a pop conditioned on the root node.

3. If the first choice in stage 1 is to STOP at the current leaf, then stage 2 and 3 are unnecessary. Hence, a choice to STOP immediately is a choice to emit another foreign word from the current English word.

4. We flatten unary transitions from the tree when computing distortion probabilities.

5. Null alignments are treated just as in the HMM model, incurring a fixed cost from any position.

This model can be simplified by removing all conditioning on node types. However, we found this variant to slightly underperform the full model described above. Intuitively, types carry information about cross-linguistic ordering preferences.

3.2 Training Approach

Because our model largely mirrors the generative process and structure of the original HMM model, we apply a nearly identical training procedure to fit the parameters to the training data via the Expectation-Maximization algorithm. Och and Ney (2003) gives a detailed exposition of the technique.

In the E-step, we employ the forward-backward algorithm and current parameters to find expected counts for each potential pair of links in each training pair. In this familiar dynamic programming approach, we must compute the distortion probabilities for each pair of English positions.

The minimal path between two leaves in a tree can be computed efficiently by first finding the path from the root to each leaf, then comparing those paths to find the nearest common ancestor and a path through it – requiring time linear in the height of the tree. Computing distortion costs independently for each pair of words in the sentence imposed a computational overhead of roughly 50% over the original HMM model. The bulk of this increase arises from the fact that distortion probabilities in this model must be computed for each unique tree, in contrast

to the original HMM which has the same distortion probabilities for all sentences of a given length.

In the M-step, we re-estimate the parameters of the model using the expected counts collected during the E-step. All of the component distributions of our lexical and distortion models are multinomials. Thus, upon assuming these expectations as values for the hidden alignment vectors, we maximize likelihood of the training data simply by computing relative frequencies for each component multinomial. For the distortion model, an expected count $c(a_j, a_{j\_})$ is allocated to all tree transitions along the path $\rho_{(a_{j\_},a_j,t)}$. These allocations are summed and normalized for each tree transition type to complete re-estimation. The method of re-estimating the lexical model remains unchanged.

Initialization of the lexical model affects performance dramatically. Using the simple but effective joint training technique of Liang et al. (2006), we initialized the model with lexical parameters from a jointly trained implementation of IBM Model 1.

3.3 Improved Posterior Inference

Liang et al. (2006) shows that thresholding the posterior probabilities of alignments improves AER relative to computing Viterbi alignments. That is, we choose a threshold τ (typically $\tau = 0.5$), and take

$$\mathbf{a} = \{(i, j) : p(a_j = i|\mathbf{f}, \mathbf{e}) > \tau\}.$$

Posterior thresholding provides computationally convenient ways to combine multiple alignments, and bidirectional combination often corrects for errors in individual directional alignment models. Liang et al. (2006) suggests a soft intersection of a model m with a reverse model r (foreign to English) that thresholds the product of their posteriors at each position:

$$\mathbf{a} = \{(i, j) : p_m(a_j = i|\mathbf{f}, \mathbf{e}) \cdot p_r(a_i = j|\mathbf{f}, \mathbf{e}) > \tau\}.$$

These intersected alignments can be quite sparse, boosting precision at the expense of recall. We explore a generalized version to this approach by varying the function c that combines p_m and p_r: $\mathbf{a} = \{(i, j) : c(p_m, p_r) > \tau\}$. If c is the $\max$ function, we recover the (hard) union of the forward and reverse posterior alignments. If c is the $\min$ function, we recover the (hard) intersection. A novel, high performing alternative is the soft union, which we evaluate in the next section:

$$c(p_m, p_r) = \frac{p_m(a_j = i|\mathbf{f}, \mathbf{e}) + p_r(a_i = j|\mathbf{f}, \mathbf{e})}{2}.$$

Syntax-alignment compatibility can be further promoted with a simple posterior decoding heuristic we call *competitive thresholding*. Given a threshold and a matrix c of combined weights for each possible link in an alignment, we include a link (i, j) only if its weight c_{ij} is above-threshold and it is connected to the maximum weighted link in both row i and column j. That is, only the maximum in each column and row and a contiguous enclosing span of above-threshold links are included in the alignment.

3.4 Related Work

This proposed model is not the first variant of the HMM model that incorporates syntax-based distortion. Lopez and Resnik (2005) considers a simpler tree distance distortion model. Daumé III and Marcu (2005) employs a syntax-aware distortion model for aligning summaries to documents, but condition upon the roots of the constituents that are jumped over during a transition, instead of those that are visited during a walk through the tree. In the case of syntactic machine translation, we want to condition on crossing constituent boundaries, even if no constituents are skipped in the process.

4 Experimental Results

To understand the behavior of this model, we computed the standard alignment error rate (AER) performance metric.[2] We also investigated extraction-specific metrics: the frequency of interior nodes – a measure of how often the alignments violate the constituent structure of English parses – and a variant of the CPER metric of Ayan and Dorr (2006).

We evaluated the performance of our model on both French-English and Chinese-English manually aligned data sets. For Chinese, we trained on the FBIS corpus and the LDC bilingual dictionary, then tested on 491 hand-aligned sentences from the 2002

[2]The hand-aligned test data has been annotated with both *sure* alignments S and *possible* alignments P, with $S \subseteq P$, according to the specifications described in Och and Ney (2003). With these alignments, we compute AER for a proposed alignment A as: $\left(1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}\right) \times 100\%$.

French	Precision	Recall	AER
Classic HMM	93.9	**93.0**	6.5
Syntactic HMM	95.2	91.5	**6.4**
GIZA++	**96.0**	86.1	8.6
Chinese	Precision	Recall	AER
Classic HMM	81.6	78.8	**19.8**
Syntactic HMM	**82.2**	76.8	20.5
GIZA++*	61.9	**82.6**	29.7

Table 1: Alignment error rates (AER) for 100k training sentences. The evaluated alignments are a soft union for French and a hard union for Chinese, both using competitive thresholding decoding. *From Ayan and Dorr (2006), *grow-diag-final* heuristic.

NIST MT evaluation set. For French, we used the Hansards data from the NAACL 2003 Shared Task.[3] We trained on 100k sentences for each language.

4.1 Alignment Error Rate

We compared our model to the original HMM model, identical in implementation to our syntactic HMM model save the distortion component. Both models were initialized using the same jointly trained Model 1 parameters (5 iterations), then trained independently for 5 iterations. Both models were then combined with an independently trained HMM model in the opposite direction: $\mathbf{f} \rightarrow \mathbf{e}$.[4] Table 1 summarizes the results; the two models perform similarly. The main benefit of our model is the effect on rule extraction, discussed below.

We also compared our French results to the public baseline GIZA++ using the script published for the NAACL 2006 Machine Translation Workshop Shared Task.[5] Similarly, we compared our Chinese results to the GIZA++ results in Ayan and Dorr (2006). Our models substantially outperform GIZA++, confirming results in Liang et al. (2006).

Table 2 shows the effect on AER of competitive thresholding and different combination functions.

[3]Following previous work, we developed our system on the 37 provided validation sentences and the first 100 sentences of the corpus test set. We used the remainder as a test set.

[4]Null emission probabilities were fixed to $\frac{1}{|\mathbf{e}|}$, inversely proportional to the length of the English sentence. The decoding threshold was held fixed at $\tau = 0.5$.

[5]Training includes 16 iterations of various IBM models and a fixed null emission probability of .01. The output of running GIZA++ in both directions was combined via intersection.

French	w/o CT	with CT
Hard Intersection (Min)	8.4	8.4
Hard Union (Max)	12.3	7.7
Soft Intersection (Product)	6.9	7.1
Soft Union (Average)	**6.7**	**6.4**
Chinese	w/o CT	with CT
Hard Intersection (Min)	27.4	27.4
Hard Union (Max)	25.0	**20.5**
Soft Intersection (Product)	25.0	25.2
Soft Union (Average)	**21.1**	21.6

Table 2: Alignment error rates (AER) by decoding method for the syntactic HMM model. The competitive thresholding heuristic (CT) is particularly helpful for the hard union combination method.

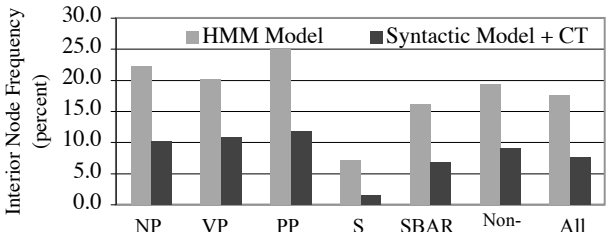
The most dramatic effect of competitive thresholding is to improve alignment quality for hard unions. It also impacts rule extraction substantially.

4.2 Rule Extraction Results

While its competitive AER certainly speaks to the potential utility of our syntactic distortion model, we proposed the model for a different purpose: to minimize the particularly troubling alignment errors that cross constituent boundaries and violate the structure of English parse trees. We found that while the HMM and Syntactic models have very similar AER, they make substantially different errors.

To investigate the differences, we measured the degree to which each set of alignments violated the supplied parse trees, by counting the frequency of interior nodes that are not null aligned. Figure 4 summarizes the results of the experiment for French: the Syntactic distortion with competitive thresholding reduces tree violations substantially. Interior node frequency is reduced by 56% overall, with the most dramatic improvement observed for clausal constituents. We observed a similar 50% reduction for the Chinese data.

Additionally, we evaluated our model with the transducer analog to the consistent phrase error rate (CPER) metric of Ayan and Dorr (2006). This evaluation computes precision, recall, and F1 of the rules extracted under a proposed alignment, relative to the rules extracted under the gold-standard sure alignments. Table 3 shows improvements in F1 by using



	NP	VP	PP	S	SBAR	Non-Terminals	All
Reduction (percent)	54.1	46.3	52.4	77.5	58.0	53.1	56.3
Corpus Frequency	14.6	10.3	6.3	4.8	1.9	41.1	100.0

Figure 4: The syntactic distortion model with competitive thresholding decreases the frequency of interior nodes for each type and the whole corpus.

the syntactic HMM model and competitive thresholding together. Individually, each of these changes contributes substantially to this increase. Together, their benefits are partially, but not fully, additive.

5 Conclusion

In light of the need to reconcile word alignments with phrase structure trees for syntactic MT, we have proposed an HMM-like model whose distortion is sensitive to such trees. Our model substantially reduces the number of interior nodes in the aligned corpus and improves rule extraction while nearly retaining the speed and alignment accuracy of the HMM model. While it remains to be seen whether these improvements impact final translation accuracy, it is reasonable to hope that, all else equal, alignments which better respect syntactic correspondences will be superior for syntactic MT.

French	Prec.	Recall	F1
Classic HMM Baseline	40.9	17.6	24.6
Syntactic HMM + CT	33.9	22.4	27.0
Relative change	-17%	27%	10%

Chinese	Prec.	Recall	F1
HMM Baseline (hard)	**66.1**	14.5	23.7
HMM Baseline (soft)	36.7	39.1	37.8
Syntactic + CT (hard)	48.0	41.6	**44.6**
Syntactic + CT (soft)	32.9	**48.7**	39.2
*Relative change**	31%	6%	18%

Table 3: Relative to the classic HMM baseline, our syntactic distortion model with competitive thresholding improves the tradeoff between precision and recall of extracted transducer rules. Both French aligners were decoded using the best-performing soft union combiner. For Chinese, we show aligners under both soft and hard union combiners. *Denotes relative change from the second line to the third line.

References

Necip Fazil Ayan and Bonnie J. Dorr. 2006. Going beyond aer: An extensive analysis of word alignments and their impact on mt. In *ACL*.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1994. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311.

Colin Cherry and Dekang Lin. 2006. Soft syntactic constraints for word alignment through discriminative training. In *ACL*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *ACL*.

Hal Daumé III and Daniel Marcu. 2005. Induction of word and phrase alignments for automatic document summarization. *Computational Linguistics*, 31(4):505–530, December.

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

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

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *HLT-NAACL*.

A. Lopez and P. Resnik. 2005. Improved hmm alignment models for languages with scarce resources. In *ACL WPT-05*.

I. Dan Melamed. 2004. Algorithms for syntax-aware statistical machine translation. In *Proceedings of the Conference on Theoretical and Methodological Issues in Machine Translation*.

Robert C. Moore. 2005. A discriminative framework for bilingual word alignment. In *EMNLP*.

Hermann Ney and Stephan Vogel. 1996. Hmm-based word alignment in statistical translation. In *COLING*.

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

Ben Taskar, Simon Lacoste-Julien, and Dan Klein. 2005. A discriminative matching approach to word alignment. In *EMNLP*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23:377–404.

Transductive learning for statistical machine translation

Nicola Ueffing
National Research Council Canada
Gatineau, QC, Canada
nicola.ueffing@nrc.gc.ca

Gholamreza Haffari and **Anoop Sarkar**
Simon Fraser University
Burnaby, BC, Canada
{ghaffar1,anoop}@cs.sfu.ca

Abstract

Statistical machine translation systems are usually trained on large amounts of bilingual text and monolingual text in the target language. In this paper we explore the use of transductive semi-supervised methods for the effective use of monolingual data from the source language in order to improve translation quality. We propose several algorithms with this aim, and present the strengths and weaknesses of each one. We present detailed experimental evaluations on the French–English EuroParl data set and on data from the NIST Chinese–English large-data track. We show a significant improvement in translation quality on both tasks.

1 Introduction

In statistical machine translation (SMT), translation is modeled as a decision process. The goal is to find the translation **t** of source sentence **s** which maximizes the posterior probability:

$$\arg\max_{\mathbf{t}} p(\mathbf{t} \mid \mathbf{s}) = \arg\max_{\mathbf{t}} p(\mathbf{s} \mid \mathbf{t}) \cdot p(\mathbf{t}) \qquad (1)$$

This decomposition of the probability yields two different statistical models which can be trained independently of each other: the translation model $p(\mathbf{s} \mid \mathbf{t})$ and the target language model $p(\mathbf{t})$.

State-of-the-art SMT systems are trained on large collections of text which consist of bilingual corpora (to learn the parameters of $p(\mathbf{s} \mid \mathbf{t})$), and of monolingual target language corpora (for $p(\mathbf{t})$). It has been shown that adding large amounts of target language text improves translation quality considerably. However, the availability of monolingual corpora in the source language does not help improve the system's performance. We will show how such corpora can be used to achieve higher translation quality.

Even if large amounts of bilingual text are given, the training of the statistical models usually suffers from sparse data. The number of possible events, i.e. phrase pairs or pairs of subtrees in the two languages, is too big to reliably estimate a probability distribution over such pairs. Another problem is that for many language pairs the amount of available bilingual text is very limited. In this work, we will address this problem and propose a general framework to solve it. Our hypothesis is that adding information from source language text can also provide improvements. Unlike adding target language text, this hypothesis is a natural semi-supervised learning problem. To tackle this problem, we propose algorithms for transductive semi-supervised learning. By transductive, we mean that we repeatedly translate sentences from the development set or test set and use the generated translations to improve the performance of the SMT system. Note that the evaluation step is still done just once at the end of our learning process. In this paper, we show that such an approach can lead to better translations despite the fact that the development and test data are typically much smaller in size than typical training data for SMT systems.

Transductive learning can be seen as a means to adapt the SMT system to a new type of text. Say a system trained on newswire is used to translate weblog texts. The proposed method adapts the trained models to the style and domain of the new input.

2 Baseline MT System

The SMT system we applied in our experiments is PORTAGE. This is a state-of-the-art phrase-based translation system which has been made available

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 25–32,
Prague, Czech Republic, June 2007.

to Canadian universities for research and education purposes. We provide a basic description here; for a detailed description see (Ueffing et al., 2007).

The models (or features) which are employed by the decoder are: (a) one or several phrase table(s), which model the translation direction $p(\mathbf{s} \mid \mathbf{t})$, (b) one or several n-gram language model(s) trained with the SRILM toolkit (Stolcke, 2002); in the experiments reported here, we used 4-gram models on the NIST data, and a trigram model on EuroParl, (c) a distortion model which assigns a penalty based on the number of source words which are skipped when generating a new target phrase, and (d) a word penalty. These different models are combined log-linearly. Their weights are optimized w.r.t. BLEU score using the algorithm described in (Och, 2003). This is done on a development corpus which we will call dev1 in this paper. The search algorithm implemented in the decoder is a dynamic-programming beam-search algorithm.

After the main decoding step, rescoring with additional models is performed. The system generates a 5,000-best list of alternative translations for each source sentence. These lists are rescored with the following models: (a) the different models used in the decoder which are described above, (b) two different features based on IBM Model 1 (Brown et al., 1993), (c) posterior probabilities for words, phrases, n-grams, and sentence length (Zens and Ney, 2006; Ueffing and Ney, 2007), all calculated over the N-best list and using the sentence probabilities which the baseline system assigns to the translation hypotheses. The weights of these additional models and of the decoder models are again optimized to maximize BLEU score. This is performed on a second development corpus, dev2.

3 The Framework

3.1 The Algorithm

Our transductive learning algorithm, Algorithm 1, is inspired by the Yarowsky algorithm (Yarowsky, 1995; Abney, 2004). The algorithm works as follows: First, the translation model is estimated based on the sentence pairs in the bilingual training data L. Then, a set of source language sentences, U, is translated based on the current model. A subset of good translations and their sources, T_i, is selected in each iteration and added to the training data. These selected sentence pairs are replaced in each iteration, and only the original bilingual training data, L, is kept fixed throughout the algorithm. The process of generating sentence pairs, selecting a subset of good sentence pairs, and updating the model is continued until a stopping condition is met. Note that we run this algorithm in a transductive setting which means that the set of sentences U is drawn either from a development set or the test set that will be used eventually to evaluate the SMT system or from additional data which is relevant to the development or test set. In Algorithm 1, changing the definition of **Estimate**, **Score** and **Select** will give us the different semi-supervised learning algorithms we will discuss in this paper.

Given the probability model $p(\mathbf{t} \mid \mathbf{s})$, consider the distribution over all possible valid translations **t** for a particular input sentence **s**. We can initialize this probability distribution to the uniform distribution for each sentence **s** in the unlabeled data U. Thus, this distribution over translations of sentences from U will have the maximum entropy. Under certain precise conditions, as described in (Abney, 2004), we can analyze Algorithm 1 as minimizing the entropy of the distribution over translations of U. However, this is true only when the functions **Estimate**, **Score** and **Select** have very prescribed definitions. In this paper, rather than analyze the convergence of Algorithm 1 we run it for a fixed number of iterations and instead focus on finding useful definitions for **Estimate**, **Score** and **Select** that can be experimentally shown to improve MT performance.

3.2 The Estimate Function

We consider the following different definitions for **Estimate** in Algorithm 1:

Full Re-training (of all translation models): If **Estimate**(L, T) estimates the model parameters based on $L \cup T$, then we have a semi-supervised algorithm that re-trains a model on the original training data L plus the sentences decoded in the last iteration. The size of L can be controlled by **filtering** the training data (see Section 3.5).

Additional Phrase Table: If, on the other hand, a new phrase translation table is learned on T only and then added as a new component in the log-linear model, we have an alternative to the full re-training

Algorithm 1 Transductive learning algorithm for statistical machine translation

Input: training set L of parallel sentence pairs. // Bilingual training data.
Input: unlabeled set U of source text. // Monolingual source language data.
Input: number of iterations R, and size of n-best list N.
$T_{-1} := \{\}$. // Additional bilingual training data.
$i := 0$. // Iteration counter.
repeat
  Training step: $\pi^{(i)} :=$ **Estimate**(L, T_{i-1}).
  $X_i := \{\}$. // The set of generated translations for this iteration.
  for sentence $\mathbf{s} \in U$ **do**
    Labeling step: Decode $\mathbf{s}$ using $\pi^{(i)}$ to obtain N best sentence pairs with their scores
    $X_i := X_i \cup \{(\mathbf{t}_n, \mathbf{s}, \pi^{(i)}(\mathbf{t}_n \,|\, \mathbf{s}))_{n=1}^{N}\}$
  end for
  Scoring step: $S_i :=$ **Score**(X_i) // Assign a score to sentence pairs $(\mathbf{t}, \mathbf{s})$ from X.
  Selection step: $T_i :=$ **Select**(X_i, S_i) // Choose a subset of *good* sentence pairs $(\mathbf{t}, \mathbf{s})$ from X.
  $i := i + 1$.
until $i > R$

of the model on labeled and unlabeled data which can be very expensive if L is very large (as on the Chinese–English data set). This additional phrase table is small and specific to the development or test set it is trained on. It overlaps with the original phrase tables, but also contains many new phrase pairs (Ueffing, 2006).

Mixture Model: Another alternative for **Estimate** is to create a mixture model of the phrase table probabilities with new phrase table probabilities

$$p(\mathbf{s} \,|\, \mathbf{t}) = \lambda \cdot L_p(\mathbf{s} \,|\, \mathbf{t}) + (1 - \lambda) \cdot T_p(\mathbf{s} \,|\, \mathbf{t}) \qquad (2)$$

where L_p and T_p are phrase table probabilities estimated on L and T, respectively. In cases where new phrase pairs are learned from T, they get added into the merged phrase table.

3.3 The Scoring Function

In Algorithm 1, the **Score** function assigns a score to each translation hypothesis **t**. We used the following scoring functions in our experiments:

Length-normalized Score: Each translated sentence pair $(\mathbf{t}, \mathbf{s})$ is scored according to the model probability $p(\mathbf{t} \,|\, \mathbf{s})$ normalized by the length $|\mathbf{t}|$ of the target sentence:

$$\textbf{Score}(\mathbf{t}, \mathbf{s}) \quad = \quad p(\mathbf{t} \,|\, \mathbf{s})^{\frac{1}{|\mathbf{t}|}} \qquad (3)$$

Confidence Estimation: The confidence estimation which we implemented follows the approaches suggested in (Blatz et al., 2003; Ueffing and Ney, 2007): The confidence score of a target sentence **t** is calculated as a log-linear combination of phrase posterior probabilities, Levenshtein-based word posterior probabilities, and a target language model score. The weights of the different scores are optimized w.r.t. classification error rate (CER).

The phrase posterior probabilities are determined by summing the sentence probabilities of all translation hypotheses in the N-best list which contain this phrase pair. The segmentation of the sentence into phrases is provided by the decoder. This sum is then normalized by the total probability mass of the N-best list. To obtain a score for the whole target sentence, the posterior probabilities of all target phrases are multiplied. The word posterior probabilities are calculated on basis of the Levenshtein alignment between the hypothesis under consideration and all other translations contained in the N-best list. For details, see (Ueffing and Ney, 2007). Again, the single values are multiplied to obtain a score for the whole sentence. For NIST, the language model score is determined using a 5-gram model trained on the English Gigaword corpus, and on French–English, we use the trigram model which was provided for the NAACL 2006 shared task.

3.4 The Selection Function

The **Select** function in Algorithm 1 is used to create the additional training data T_i which will be used in

the next iteration $i + 1$ by **Estimate** to augment the original bilingual training data. We use the following selection functions:

Importance Sampling: For each sentence **s** in the set of unlabeled sentences U, the Labeling step in Algorithm 1 generates an N-best list of translations, and the subsequent Scoring step assigns a score for each translation **t** in this list. The set of generated translations for all sentences in U is the event space and the scores are used to put a probability distribution over this space, simply by renormalizing the scores described in Section 3.3. We use importance sampling to select K translations from this distribution. Sampling is done with replacement which means that the same translation may be chosen several times. These K sampled translations and their associated source sentences make up the additional training data T_i.

Selection using a Threshold: This method compares the score of each single-best translation to a threshold. The translation is considered reliable and added to the set T_i if its score exceeds the threshold. Else it is discarded and not used in the additional training data. The threshold is optimized on the development beforehand. Since the scores of the translations change in each iteration, the size of T_i also changes.

Keep All: This method does not perform any filtering at all. It is simply assumed that all translations in the set X_i are reliable, and none of them are discarded. Thus, in each iteration, the result of the selection step will be $T_i = X_i$. This method was implemented mainly for comparison with other selection methods.

3.5 Filtering the Training Data

In general, having more training data improves the quality of the trained models. However, when it comes to the translation of a particular test set, the question is whether *all* of the available training data are relevant to the translation task or not. Moreover, working with large amounts of training data requires more computational power. So if we can identify a subset of training data which are relevant to the current task and use only this to re-train the models, we can reduce computational complexity significantly.

We propose to **Filter** the training data, either bilingual or monolingual text, to identify the parts which are relevant w.r.t. the test set. This filtering is based on n-gram coverage. For a source sentence **s** in the training data, its n-gram coverage over the sentences in the test set is computed. The average over several n-gram lengths is used as a measure of relevance of this training sentence w.r.t. the test corpus. Based on this, we select the top K source sentences or sentence pairs.

corpus	use	sentences
EuroParl	phrase table+LM	688K
train100k	phrase table	100K
train150k	phrase table	150K
dev06	dev1	2,000
test06	test	3,064

Table 1: French–English corpora

corpus	use	sentences
non-UN	phrase table+LM	3.2M
UN	phrase table+LM	5.0M
English Gigaword	LM	11.7M
multi-p3	dev1	935
multi-p4	dev2	919
eval-04	test	1,788
eval-06	test	3,940

Table 2: NIST Chinese–English corpora

4 Experimental Results

4.1 Setting

We ran experiments on two different corpora: one is the French–English translation task from the EuroParl corpus, and the other one is Chinese–English translation as performed in the NIST MT evaluation (www.nist.gov/speech/tests/mt).

For the French–English translation task, we used the EuroParl corpus as distributed for the shared task in the NAACL 2006 workshop on statistical machine translation. The corpus statistics are shown in Table 1. Furthermore we filtered the EuroParl corpus, as explained in Section 3.5, to create two smaller bilingual corpora (train100k and train150k in Table 1). The development set is used to optimize the model weights in the decoder, and the evaluation is done on the test set provided for the NAACL 2006 shared task.

For the Chinese–English translation task, we used the corpora distributed for the large-data track in the

setting		EuroParl	NIST
full re-training w/ filtering		$*$	$**$
full re-training		$**$	†
mixture model		$*$	†
new phrase table ff:			
keep all		$**$	$*$
imp. sampling	norm.	$**$	$*$
	conf.	$**$	$*$
threshold	norm.	$**$	$*$
	conf.	$**$	$*$

Table 3: Feasibility of settings for Algorithm 1

2006 NIST evaluation (see Table 2). We used the LDC segmenter for Chinese. The multiple translation corpora multi-p3 and multi-p4 were used as development corpora. Evaluation was performed on the 2004 and 2006 test sets. Note that the training data consists mainly of written text, whereas the test sets comprise three and four different genres: editorials, newswire and political speeches in the 2004 test set, and broadcast conversations, broadcast news, newsgroups and newswire in the 2006 test set. Most of these domains have characteristics which are different from those of the training data, e.g., broadcast conversations have characteristics of spontaneous speech, and the newsgroup data is comparatively unstructured.

Given the particular data sets described above, Table 3 shows the various options for the **Estimate**, **Score** and **Select** functions (see Section 3). The table provides a quick guide to the experiments we present in this paper vs. those we did not attempt due to computational infeasibility. We ran experiments corresponding to all entries marked with $*$ (see Section 4.2). For those marked $**$ the experiments produced only minimal improvement over the baseline and so we do not discuss them in this paper. The entries marked as † were not attempted because they are not feasible (e.g. full re-training on the NIST data). However, these were run on the smaller EuroParl corpus.

Evaluation Metrics

We evaluated the generated translations using three different evaluation metrics: BLEU score (Papineni et al., 2002), mWER (multi-reference word error rate), and mPER (multi-reference position-independent word error rate) (Nießen et al., 2000). Note that BLEU score measures quality, whereas mWER and mPER measure translation errors. We will present 95%-confidence intervals for the baseline system which are calculated using bootstrap resampling. The metrics are calculated w.r.t. one and four English references: the EuroParl data comes with one reference, the NIST 2004 evaluation set and the NIST section of the 2006 evaluation set are provided with four references each, whereas the GALE section of the 2006 evaluation set comes with one reference only. This results in much lower BLEU scores and higher error rates for the translations of the GALE set (see Section 4.2). Note that these values do not indicate lower translation quality, but are simply a result of using only one reference.

4.2 Results

EuroParl

We ran our initial experiments on EuroParl to explore the behavior of the transductive learning algorithm. In all experiments reported in this subsection, the test set was used as unlabeled data. The selection and scoring was carried out using importance sampling with normalized scores. In one set of experiments, we used the 100K and 150K training sentences filtered according to n-gram coverage over the test set. We fully re-trained the phrase tables on these data and 8,000 test sentence pairs sampled from 20-best lists in each iteration. The results on the test set can be seen in Figure 1. The BLEU score increases, although with slight variation, over the iterations. In total, it increases from 24.1 to 24.4 for the 100K filtered corpus, and from 24.5 to 24.8 for 150K, respectively. Moreover, we see that the BLEU score of the system using 100K training sentence pairs and transductive learning is the same as that of the one trained on 150K sentence pairs. So the information extracted from untranslated test sentences is equivalent to having an additional 50K sentence pairs.

In a second set of experiments, we used the whole EuroParl corpus and the sampled sentences for fully re-training the phrase tables in each iteration. We ran the algorithm for three iterations and the BLEU score increased from 25.3 to 25.6. Even though this

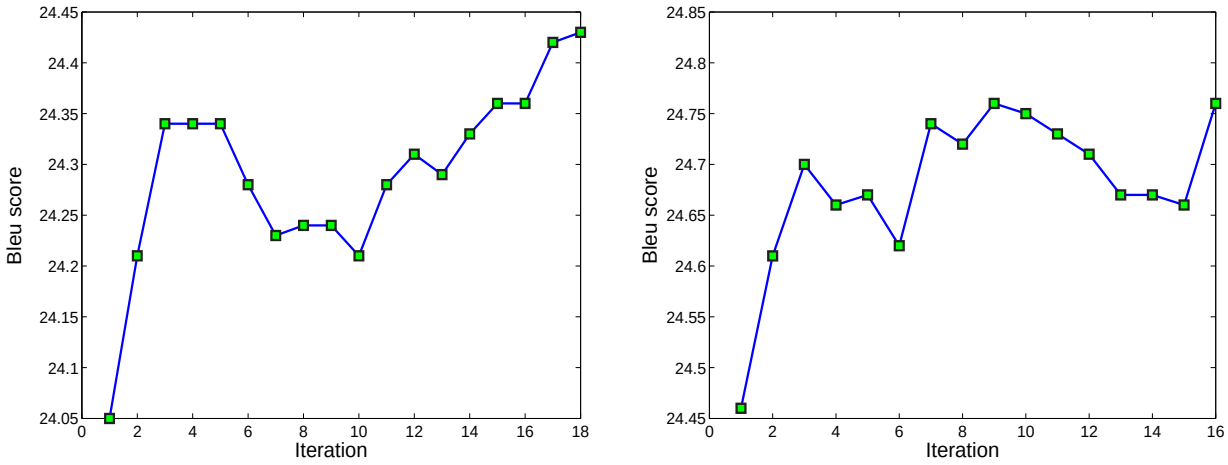


Figure 1: Translation quality for importance sampling with full re-training on train100k (left) and train150k (right). EuroParl French–English task.

is a small increase, it shows that the unlabeled data contains some information which can be explored in transductive learning.

In a third experiment, we applied the mixture model idea as explained in Section 3.2. The initially learned phrase table was merged with the learned phrase table in each iteration with a weight of $\lambda = 0.1$. This value for λ was found based on cross validation on a development set. We ran the algorithm for 20 iterations and BLEU score increased from 25.3 to 25.7. Since this is very similar to the result obtained with the previous method, but with an additional parameter λ to optimize, we did not use mixture models on NIST.

Note that the single improvements achieved here are slightly below the 95%-significance level. However, we observe them consistently in all settings.

NIST

Table 4 presents translation results on NIST with different versions of the scoring and selection methods introduced in Section 3. In these experiments, the unlabeled data U for Algorithm 1 is the development or test corpus. For this corpus U, 5,000-best lists were generated using the baseline SMT system. Since re-training the full phrase tables is not feasible here, a (small) additional phrase table, specific to U, was trained and plugged into the SMT system as an additional model. The decoder weights thus had to be optimized again to determine the appropriate weight for this new phrase table. This was done on the dev1 corpus, using the phrase table specific to dev1. Every time a new corpus is to be translated, an adapted phrase table is created using transductive learning and used with the weight which has been learned on dev1. In the first experiment presented in Table 4, all of the generated 1-best translations were kept and used for training the adapted phrase tables. This method yields slightly higher translation quality than the baseline system. The second approach we studied is the use of importance sampling (IS) over 20-best lists, based either on length-normalized sentence scores (norm.) or confidence scores (conf.). As the results in Table 4 show, both variants outperform the first method, with a consistent improvement over the baseline across all test corpora and evaluation metrics. The third method uses a threshold-based selection method. Combined with confidence estimation as scoring method, this yields the best results. All improvements over the baseline are significant at the 95%-level.

Table 5 shows the translation quality achieved on the NIST test sets when additional source language data from the Chinese Gigaword corpus comprising newswire text is used for transductive learning. These Chinese sentences were sorted according to their n-gram overlap (see Section 3.5) with the development corpus, and the top 5,000 Chinese sentences were used. The selection and scoring in Algorithm 1 were performed using confidence estimation with a threshold. Again, a new phrase table was trained on these data. As can be seen in Table 5, this

select	score	BLEU[%]	mWER[%]	mPER[%]
eval-04 (4 refs.)				
baseline		31.8±0.7	66.8±0.7	41.5±0.5
keep all		33.1	66.0	41.3
IS	norm.	**33.5**	65.8	40.9
	conf.	33.2	65.6	**40.4**
thr	norm.	**33.5**	65.9	40.8
	conf.	**33.5**	**65.3**	40.8
eval-06 GALE (1 ref.)				
baseline		12.7±0.5	75.8±0.6	54.6±0.6
keep all		12.9	75.7	55.0
IS	norm.	13.2	74.7	54.1
	conf.	12.9	74.4	53.5
thr	norm.	12.7	75.2	54.2
	conf.	**13.6**	**73.4**	**53.2**
eval-06 NIST (4 refs.)				
baseline		27.9±0.7	67.2±0.6	44.0±0.5
keep all		28.1	66.5	44.2
IS	norm.	28.7	66.1	43.6
	conf.	28.4	65.8	**43.2**
thr	norm.	28.3	66.1	43.5
	conf.	**29.3**	**65.6**	**43.2**

Table 4: Translation quality using an additional adapted phrase table trained on the dev/test sets. Different selection and scoring methods. NIST Chinese–English, best results printed in boldface.

system outperforms the baseline system on all test corpora. The error rates are significantly reduced in all three settings, and BLEU score increases in all cases. A comparison with Table 4 shows that transductive learning on the development set and test corpora, adapting the system to their domain and style, is more effective in improving the SMT system than the use of additional source language data.

In all experiments on NIST, Algorithm 1 was run for one iteration. We also investigated the use of an iterative procedure here, but this did not yield any improvement in translation quality.

5 Previous Work

Semi-supervised learning has been previously applied to improve word alignments. In (Callison-Burch et al., 2004), a generative model for word alignment is trained using unsupervised learning on parallel text. In addition, another model is trained on a small amount of hand-annotated word alignment data. A mixture model provides a probability for word alignment. Experiments showed that putting a large weight on the model trained on labeled data performs best. Along similar lines, (Fraser and Marcu, 2006) combine a generative model of word alignment with a log-linear discriminative model trained on a small set of hand aligned sentences. The word alignments are used to train a standard phrase-based SMT system, resulting in increased translation quality .

system	BLEU[%]	mWER[%]	mPER[%]
eval-04 (4 refs.)			
baseline	31.8±0.7	66.8±0.7	41.5±0.5
add Chin. data	32.8	65.7	40.9
eval-06 GALE (1 ref.)			
baseline	12.7±0.5	75.8±0.6	54.6±0.6
add Chin. data	13.1	73.9	53.5
eval-06 NIST (4 refs.)			
baseline	27.9±0.7	67.2±0.6	44.0±0.5
add Chin. data	28.1	65.8	43.2

Table 5: Translation quality using an additional phrase table trained on monolingual Chinese news data. Selection step using threshold on confidence scores. NIST Chinese–English.

In (Callison-Burch, 2002) co-training is applied to MT. This approach requires several source languages which are sentence-aligned with each other and all translate into the same target language. One language pair creates data for another language pair and can be naturally used in a (Blum and Mitchell, 1998)-style co-training algorithm. Experiments on the EuroParl corpus show a decrease in WER. However, the selection algorithm applied there is actually supervised because it takes the reference translation into account. Moreover, when the algorithm is run long enough, large amounts of co-trained data injected too much noise and performance degraded.

Self-training for SMT was proposed in (Ueffing, 2006). An existing SMT system is used to translate the development or test corpus. Among the generated machine translations, the reliable ones are automatically identified using thresholding on confidence scores. The work which we presented here differs from (Ueffing, 2006) as follows:

- We investigated different ways of scoring and selecting the reliable translations and compared our method to this work. In addition to the con-

fidence estimation used there, we applied importance sampling and combined it with confidence estimation for transductive learning.

- We studied additional ways of exploring the newly created bilingual data, namely re-training the full phrase translation model or creating a mixture model.
- We proposed an iterative procedure which translates the monolingual source language data anew in each iteration and then re-trains the phrase translation model.
- We showed how additional monolingual source-language data can be used in transductive learning to improve the SMT system.

6 Discussion

It is not intuitively clear why the SMT system can learn something from its own output and is improved through semi-supervised learning. There are two main reasons for this improvement: Firstly, the selection step provides important feedback for the system. The confidence estimation, for example, discards translations with low language model scores or posterior probabilities. The selection step discards bad machine translations and reinforces phrases of high quality. As a result, the probabilities of low-quality phrase pairs, such as noise in the table or overly confident singletons, degrade. Our experiments comparing the various settings for transductive learning shows that selection clearly outperforms the method which keeps all generated translations as additional training data. The selection methods investigated here have been shown to be well-suited to boost the performance of semi-supervised learning for SMT.

Secondly, our algorithm constitutes a way of adapting the SMT system to a new domain or style without requiring bilingual training or development data. Those phrases in the existing phrase tables which are relevant for translating the new data are reinforced. The probability distribution over the phrase pairs thus gets more focused on the (reliable) parts which are relevant for the test data. For an analysis of the self-trained phrase tables, examples of translated sentences, and the phrases used in translation, see (Ueffing, 2006).

References

S. Abney. 2004. Understanding the Yarowsky Algorithm. *Comput. Ling.*, 30(3).

J. Blatz, E. Fitzgerald, G. Foster, S. Gandrabur, C. Goutte, A. Kulesza, A. Sanchis, and N. Ueffing. 2003. Confidence estimation for machine translation. Final report, JHU/CLSP Summer Workshop. `www.clsp.jhu.edu/ws2003/groups/estimate/`.

A. Blum and T. Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-Training. In *Proc. Computational Learning Theory*.

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2).

C. Callison-Burch, D. Talbot, and M. Osborne. 2004. Statistical machine translation with word- and sentence-aligned parallel corpora. In *Proc. ACL*.

C. Callison-Burch. 2002. Co-training for statistical machine translation. Master's thesis, School of Informatics, University of Edinburgh.

A. Fraser and D. Marcu. 2006. Semi-supervised training for statistical word alignment. In *Proc. ACL*.

S. Nießen, F. J. Och, G. Leusch, and H. Ney. 2000. An evaluation tool for machine translation: Fast evaluation for MT research. In *Proc. LREC*.

F. J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. ACL*.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. ACL*.

A. Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proc. ICSLP*.

N. Ueffing and H. Ney. 2007. Word-level confidence estimation for machine translation. *Computational Linguistics*, 33(1):9–40.

N. Ueffing, M. Simard, S. Larkin, and J. H. Johnson. 2007. NRC's Portage system for WMT 2007. In *Proc. ACL Workshop on SMT*.

N. Ueffing. 2006. Using monolingual source-language data to improve MT performance. In *Proc. IWSLT*.

D. Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proc. ACL*.

R. Zens and H. Ney. 2006. N-gram posterior probabilities for statistical machine translation. In *Proc. HLT/NAACL Workshop on SMT*.

Word Sense Disambiguation Improves Statistical Machine Translation

Yee Seng Chan and **Hwee Tou Ng**
Department of Computer Science
National University of Singapore
3 Science Drive 2
Singapore 117543
{chanys, nght}@comp.nus.edu.sg

David Chiang
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292, USA
chiang@isi.edu

Abstract

Recent research presents conflicting evidence on whether word sense disambiguation (WSD) systems can help to improve the performance of statistical machine translation (MT) systems. In this paper, we successfully integrate a state-of-the-art WSD system into a state-of-the-art hierarchical phrase-based MT system, Hiero. We show for the first time that integrating a WSD system improves the performance of a state-of-the-art statistical MT system on an actual translation task. Furthermore, the improvement is statistically significant.

1 Introduction

Many words have multiple meanings, depending on the context in which they are used. Word sense disambiguation (WSD) is the task of determining the correct meaning or sense of a word in context. WSD is regarded as an important research problem and is assumed to be helpful for applications such as machine translation (MT) and information retrieval.

In translation, different senses of a word *w* in a source language may have different translations in a target language, depending on the particular meaning of *w* in context. Hence, the assumption is that in resolving sense ambiguity, a WSD system will be able to help an MT system to determine the correct translation for an ambiguous word. To determine the correct sense of a word, WSD systems typically use a wide array of features that are not limited to the local context of *w*, and some of these features may not be used by state-of-the-art statistical MT systems.

To perform translation, state-of-the-art MT systems use a statistical phrase-based approach (Marcu and Wong, 2002; Koehn et al., 2003; Och and Ney, 2004) by treating phrases as the basic units of translation. In this approach, a phrase can be any sequence of consecutive words and is not necessarily linguistically meaningful. Capitalizing on the strength of the phrase-based approach, Chiang (2005) introduced a *hierarchical* phrase-based statistical MT system, Hiero, which achieves significantly better translation performance than Pharaoh (Koehn, 2004a), which is a state-of-the-art phrase-based statistical MT system.

Recently, some researchers investigated whether performing WSD will help to improve the performance of an MT system. Carpuat and Wu (2005) integrated the translation predictions from a Chinese WSD system (Carpuat et al., 2004) into a Chinese-English word-based statistical MT system using the ISI ReWrite decoder (Germann, 2003). Though they acknowledged that directly using English translations as word senses would be ideal, they instead predicted the HowNet sense of a word and then used the English gloss of the HowNet sense as the WSD model's predicted translation. They did not incorporate their WSD model or its predictions into their translation model; rather, they used the WSD predictions either to constrain the options available to their decoder, or to postedit the output of their decoder. They reported the negative result that WSD decreased the performance of MT based on their experiments.

In another work (Vickrey et al., 2005), the WSD problem was recast as a *word translation* task. The

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 33–40,
Prague, Czech Republic, June 2007.

translation choices for a word *w* were defined as the set of words or phrases aligned to *w*, as gathered from a word-aligned parallel corpus. The authors showed that they were able to improve their model's accuracy on two simplified translation tasks: word translation and blank-filling.

Recently, Cabezas and Resnik (2005) experimented with incorporating WSD translations into Pharaoh, a state-of-the-art phrase-based MT system (Koehn et al., 2003). Their WSD system provided additional translations to the phrase table of Pharaoh, which fired a new model feature, so that the decoder could weigh the additional alternative translations against its own. However, they could not automatically tune the weight of this feature in the same way as the others. They obtained a relatively small improvement, and no statistical significance test was reported to determine if the improvement was statistically significant.

Note that the experiments in (Carpuat and Wu, 2005) did not use a state-of-the-art MT system, while the experiments in (Vickrey et al., 2005) were not done using a full-fledged MT system and the evaluation was not on how well each source sentence was translated as a whole. The relatively small improvement reported by Cabezas and Resnik (2005) without a statistical significance test appears to be inconclusive. Considering the conflicting results reported by prior work, it is not clear whether a WSD system can help to improve the performance of a state-of-the-art statistical MT system.

In this paper, we successfully integrate a state-of-the-art WSD system into the state-of-the-art hierarchical phrase-based MT system, Hiero (Chiang, 2005). The integration is accomplished by introducing two additional features into the MT model which operate on the existing rules of the grammar, without introducing competing rules. These features are treated, both in feature-weight tuning and in decoding, on the same footing as the rest of the model, allowing it to weigh the WSD model predictions against other pieces of evidence so as to optimize translation accuracy (as measured by BLEU). The contribution of our work lies in showing for the first time that integrating a WSD system significantly improves the performance of a state-of-the-art statistical MT system on an actual translation task.

In the next section, we describe our WSD system. Then, in Section 3, we describe the Hiero MT system and introduce the two new features used to integrate the WSD system into Hiero. In Section 4, we describe the training data used by the WSD system. In Section 5, we describe how the WSD translations provided are used by the decoder of the MT system. In Section 6 and 7, we present and analyze our experimental results, before concluding in Section 8.

2 Word Sense Disambiguation

Prior research has shown that using Support Vector Machines (SVM) as the learning algorithm for WSD achieves good results (Lee and Ng, 2002). For our experiments, we use the SVM implementation of (Chang and Lin, 2001) as it is able to work on multi-class problems to output the classification probability for each class.

Our implemented WSD classifier uses the knowledge sources of local collocations, parts-of-speech (POS), and surrounding words, following the successful approach of (Lee and Ng, 2002). For local collocations, we use 3 features, $w_{-1}w_{+1}$, w_{-1}, and w_{+1}, where w_{-1} (w_{+1}) is the token immediately to the left (right) of the current ambiguous word occurrence w. For parts-of-speech, we use 3 features, P_{-1}, P_0, and P_{+1}, where P_0 is the POS of w, and P_{-1} (P_{+1}) is the POS of w_{-1} (w_{+1}). For surrounding words, we consider all unigrams (single words) in the surrounding context of *w*. These unigrams can be in a different sentence from *w*. We perform feature selection on surrounding words by including a unigram only if it occurs 3 or more times in some sense of *w* in the training data.

To measure the accuracy of our WSD classifier, we evaluate it on the test data of SENSEVAL-3 Chinese lexical-sample task. We obtain accuracy that compares favorably to the best participating system in the task (Carpuat et al., 2004).

3 Hiero

Hiero (Chiang, 2005) is a hierarchical phrase-based model for statistical machine translation, based on weighted synchronous context-free grammar (CFG) (Lewis and Stearns, 1968). A synchronous CFG consists of rewrite rules such as the following:

$$\mathrm{X} \rightarrow \langle \gamma, \alpha \rangle \quad (1)$$

where X is a non-terminal symbol, γ (α) is a string of terminal and non-terminal symbols in the source (target) language, and there is a one-to-one correspondence between the non-terminals in γ and α indicated by co-indexation. Hence, γ and α always have the same number of non-terminal symbols. For instance, we could have the following grammar rule:

$$\text{X} \rightarrow \langle \text{每 月 到 X}_{\boxed{1}}, \text{go to X}_{\boxed{1}} \text{ every month to} \rangle \quad (2)$$

where boxed indices represent the correspondences between non-terminal symbols.

Hiero extracts the synchronous CFG rules automatically from a word-aligned parallel corpus. To translate a source sentence, the goal is to find its most probable derivation using the extracted grammar rules. Hiero uses a general log-linear model (Och and Ney, 2002) where the weight of a derivation D for a particular source sentence and its translation is

$$w(D) = \prod_i \phi_i(D)^{\lambda_i} \quad (3)$$

where ϕ_i is a feature function and λ_i is the weight for feature ϕ_i. To ensure efficient decoding, the ϕ_i are subject to certain locality restrictions. Essentially, they should be defined as products of functions defined on isolated synchronous CGF rules; however, it is possible to extend the domain of locality of the features somewhat. A *n*-gram language model adds a dependence on ($n-1$) neighboring target-side words (Wu, 1996; Chiang, 2007), making decoding much more difficult but still polynomial; in this paper, we add features that depend on the neighboring *source-side* words, which does not affect decoding complexity at all because the source string is fixed. In principle we could add features that depend on arbitrary source-side context.

3.1 New Features in Hiero for WSD

To incorporate WSD into Hiero, we use the translations proposed by the WSD system to help Hiero obtain a better or more probable derivation during the translation of each source sentence. To achieve this, when a grammar rule *R* is considered during decoding, and we recognize that some of the terminal symbols (words) in α are also chosen by the WSD system as translations for some terminal symbols (words) in γ, we compute the following features:

- $P_{wsd}(t \mid s)$ gives the contextual probability of the WSD classifier choosing t as a translation for s, where t (s) is some substring of terminal symbols in α (γ). Because this probability only applies to some rules, and we don't want to penalize those rules, we must add another feature,

- $Pty_{wsd} = \exp(-|t|)$, where t is the translation chosen by the WSD system. This feature, with a negative weight, rewards rules that use translations suggested by the WSD module.

Note that we can take the negative logarithm of the rule/derivation weights and think of them as costs rather than probabilities.

4 Gathering Training Examples for WSD

Our experiments were for Chinese to English translation. Hence, in the context of our work, a synchronous CFG grammar rule $\text{X} \rightarrow \langle \gamma, \alpha \rangle$ gathered by Hiero consists of a Chinese portion γ and a corresponding English portion α, where each portion is a sequence of words and non-terminal symbols.

Our WSD classifier suggests a list of English phrases (where each phrase consists of one or more English words) with associated contextual probabilities as possible translations for each particular Chinese phrase. In general, the Chinese phrase may consist of *k* Chinese words, where $k = 1, 2, 3, \ldots$. However, we limit k to 1 or 2 for experiments reported in this paper. Future work can explore enlarging *k*.

Whenever Hiero is about to extract a grammar rule where its Chinese portion is a phrase of one or two Chinese words with no non-terminal symbols, we note the location (sentence and token offset) in the Chinese half of the parallel corpus from which the Chinese portion of the rule is extracted. The actual sentence in the corpus containing the Chinese phrase, and the one sentence before and the one sentence after that actual sentence, will serve as the context for one training example for the Chinese phrase, with the corresponding English phrase of the grammar rule as its translation. Hence, unlike traditional WSD where the sense classes are tied to a specific sense inventory, our "senses" here consist of the English phrases extracted as translations for each Chinese phrase. Since the extracted training data may

be noisy, for each Chinese phrase, we remove English translations that occur only once. Furthermore, we only attempt WSD classification for those Chinese phrases with at least 10 training examples.

Using the WSD classifier described in Section 2, we classified the words in each Chinese source sentence to be translated. We first performed WSD on all single Chinese words which are either noun, verb, or adjective. Next, we classified the Chinese phrases consisting of 2 consecutive Chinese words by simply treating the phrase as *a single unit*. When performing classification, we give as output the set of English translations with associated context-dependent probabilities, which are the probabilities of a Chinese word (phrase) translating into each English phrase, depending on the context of the Chinese word (phrase). After WSD, the *i*th word c_i in every Chinese sentence may have up to 3 sets of associated translations provided by the WSD system: a set of translations for c_i as a single word, a second set of translations for $c_{i-1}c_i$ considered as a single unit, and a third set of translations for c_ic_{i+1} considered as a single unit.

5 Incorporating WSD during Decoding

The following tasks are done for each rule that is considered during decoding:

- identify Chinese words to suggest translations for
- match suggested translations against the English side of the rule
- compute features for the rule

The WSD system is able to predict translations only for a subset of Chinese words or phrases. Hence, we must first identify which parts of the Chinese side of the rule have suggested translations available. Here, we consider substrings of length up to two, and we give priority to longer substrings.

Next, we want to know, for each Chinese substring considered, whether the WSD system supports the Chinese-English translation represented by the rule. If the rule is finally chosen as part of the best derivation for translating the Chinese sentence, then all the words in the English side of the rule will appear in the translated English sentence. Hence, we need to match the translations suggested by the WSD system against the English side of the rule. It is for these matching rules that the WSD features will apply.

The translations proposed by the WSD system may be more than one word long. In order for a proposed translation to match the rule, we require two conditions. First, the proposed translation must be a substring of the English side of the rule. For example, the proposed translation "every to" would not match the chunk "every month to". Second, the match must contain at least one aligned Chinese-English word pair, but we do not make any other requirements about the alignment of the other Chinese or English words.[1] If there are multiple possible matches, we choose the longest proposed translation; in the case of a tie, we choose the proposed translation with the highest score according to the WSD model.

Define a *chunk* of a rule to be a maximal substring of terminal symbols on the English side of the rule. For example, in Rule (2), the chunks would be "go to" and "every month to". Whenever we find a matching WSD translation, we mark the whole chunk on the English side as consumed.

Finally, we compute the feature values for the rule. The feature $P_{wsd}(t \mid s)$ is the sum of the costs (according to the WSD model) of all the matched translations, and the feature Pty_{wsd} is the sum of the lengths of all the matched translations.

Figure 1 shows the pseudocode for the rule scoring algorithm in more detail, particularly with regards to resolving conflicts between overlapping matches. To illustrate the algorithm given in Figure 1, consider Rule (2). Hereafter, we will use symbols to represent the Chinese and English words in the rule: c_1, c_2, and c_3 will represent the Chinese words "每", "月", and "到" respectively. Similarly, e_1, e_2, e_3, e_4, and e_5 will represent the English words *go*, *to*, *every*, *month*, and *to* respectively. Hence, Rule (2) has two chunks: e_1e_2 and $e_3e_4e_5$. When the rule is extracted from the parallel corpus, it has these alignments between the words of its Chinese and English portion: $\{c_1$–e_3,c_2–e_4,c_3–e_1,c_3–e_2,c_3–$e_5\}$, which means that c_1 is aligned to e_3, c_2 is aligned to

[1] In order to check this requirement, we extended Hiero to make word alignment information available to the decoder.

Input: rule *R* considered during decoding with its own associated $cost_R$
L_c = list of symbols in Chinese portion of *R*
WSDcost = 0
i = 1
while i $\leq$ len(L_c):
    c_i = *i*th symbol in L_c
    if c_i is a Chinese word (i.e., not a non-terminal symbol):
        seenChunk = $\emptyset$ // seenChunk is a global variable and is passed by reference to matchWSD
        if (c_i is not the last symbol in L_c) and (c_{i+1} is a terminal symbol): then c_{i+1}=(*i*+1)th symbol in L_c, else c_{i+1} = NULL
        if (c_{i+1}!=NULL) and (c_i, c_{i+1}) as a *single unit* has WSD translations:
            WSD_c = set of WSD translations for (c_i, c_{i+1}) as a single unit with context-dependent probabilities
            WSDcost = WSDcost + matchWSD(c_i, WSD_c, seenChunk)
            WSDcost = WSDcost + matchWSD(c_{i+1}, WSD_c, seenChunk)
            i = i + 1
        else:
            WSD_c = set of WSD translations for c_i with context-dependent probabilities
            WSDcost = WSDcost + matchWSD(c_i, WSD_c, seenChunk)
    i = i + 1
$cost_R$ = $cost_R$ + WSDcost

matchWSD(*c*, WSD_c, seenChunk):
    // seenChunk is the set of chunks of *R* already examined for possible matching WSD translations
    cost = 0
    ChunkSet = set of chunks in *R* aligned to *c*
    for $chunk_j$ in ChunkSet:
        if $chunk_j$ not in seenChunk:
            seenChunk = seenChunk $\cup$ { $chunk_j$ }
            E_{chunk_j} = set of English words in $chunk_j$ aligned to *c*
            $Candidate_{wsd}$ = $\emptyset$
            for wsd_k in WSD_c:
                if (wsd_k is sub-sequence of $chunk_j$) and (wsd_k contains at least one word in E_{chunk_j})
                    $Candidate_{wsd}$ = $Candidate_{wsd}$ $\cup$ { wsd_k }
            wsd_{best} = *best matching* translation in $Candidate_{wsd}$ against $chunk_j$
            cost = cost + costByWSDfeatures(wsd_{best}) // costByWSDfeatures sums up the cost of the two WSD features
    return cost

Figure 1: WSD translations affecting the cost of a rule *R* considered during decoding.

e_4, and c_3 is aligned to e_1, e_2, and e_5. Although all words are aligned here, in general for a rule, some of its Chinese or English words may not be associated with any alignments.

In our experiment, c_1c_2 as a phrase has a list of translations proposed by the WSD system, including the English phrase "every month". *matchWSD* will first be invoked for c_1, which is aligned to only one chunk $e_3e_4e_5$ via its alignment with e_3. Since "every month" is a sub-sequence of the chunk and also contains the word e_3 ("every"), it is noted as a candidate translation. Later, it is determined that the most number of words any candidate translation has is two words. Since among all the 2-word candidate translations, the translation "every month" has the highest translation probability as assigned by the WSD classifier, it is chosen as the best matching translation for the chunk. *matchWSD* is then invoked for c_2, which is aligned to only one chunk $e_3e_4e_5$. However, since this chunk has already been examined by c_1 with which it is considered as a phrase, no further matching is done for c_2. Next, *matchWSD* is invoked for c_3, which is aligned to both chunks of *R*. The English phrases "go to" and "to" are among the list of translations proposed by the WSD system for c_3, and they are eventually chosen as the best matching translations for the chunks e_1e_2 and $e_3e_4e_5$, respectively.

6 Experiments

As mentioned, our experiments were on Chinese to English translation. Similar to (Chiang, 2005), we trained the Hiero system on the FBIS corpus, used the NIST MT 2002 evaluation test set as our development set to tune the feature weights, and the NIST MT 2003 evaluation test set as our test data. Using

System	BLEU-4	Individual *n*-gram precisions			
		1	2	3	4
Hiero	29.73	74.73	40.14	21.83	11.93
Hiero+WSD	30.30	74.82	40.40	22.45	12.42

Table 1: BLEU scores

System	Features									
	$P_{lm}(e)$	$P(\gamma\|\alpha)$	$P(\alpha\|\gamma)$	$P_w(\gamma\|\alpha)$	$P_w(\alpha\|\gamma)$	Pty_{phr}	$Glue$	Pty_{word}	$P_{wsd}(t\|s)$	Pty_{wsd}
Hiero	0.2337	0.0882	0.1666	0.0393	0.1357	0.0665	−0.0582	−0.4806	-	-
Hiero+WSD	0.1937	0.0770	0.1124	0.0487	0.0380	0.0988	−0.0305	−0.1747	0.1051	−0.1611

Table 2: Weights for each feature obtained by MERT training. The first eight features are those used by Hiero in (Chiang, 2005).

the English portion of the FBIS corpus and the Xinhua portion of the Gigaword corpus, we trained a trigram language model using the SRI Language Modelling Toolkit (Stolcke, 2002). Following (Chiang, 2005), we used the version 11a NIST BLEU script with its default settings to calculate the BLEU scores (Papineni et al., 2002) based on case-insensitive *n*-gram matching, where *n* is up to 4.

First, we performed word alignment on the FBIS parallel corpus using GIZA++ (Och and Ney, 2000) in both directions. The word alignments of both directions are then combined into a single set of alignments using the "diag-and" method of Koehn et al. (2003). Based on these alignments, synchronous CFG rules are then extracted from the corpus. While Hiero is extracting grammar rules, we gathered WSD training data by following the procedure described in section 4.

6.1 Hiero Results

Using the MT 2002 test set, we ran the minimum-error rate training (MERT) (Och, 2003) with the decoder to tune the weights for each feature. The weights obtained are shown in the row *Hiero* of Table 2. Using these weights, we run Hiero's decoder to perform the actual translation of the MT 2003 test sentences and obtained a BLEU score of 29.73, as shown in the row *Hiero* of Table 1. This is higher than the score of 28.77 reported in (Chiang, 2005), perhaps due to differences in word segmentation, etc. Note that comparing with the MT systems used in (Carpuat and Wu, 2005) and (Cabezas and Resnik, 2005), the Hiero system we are using represents a much stronger baseline MT system upon which the WSD system must improve.

6.2 Hiero+WSD Results

We then added the WSD features of Section 3.1 into Hiero and reran the experiment. The weights obtained by MERT are shown in the row *Hiero+WSD* of Table 2. We note that a negative weight is learnt for Pty_{wsd}. This means that in general, the model prefers grammar rules having chunks that matches WSD translations. This matches our intuition. Using the weights obtained, we translated the test sentences and obtained a BLEU score of **30.30**, as shown in the row *Hiero+WSD* of Table 1. The improvement of 0.57 is statistically significant at $p < 0.05$ using the sign-test as described by Collins et al. (2005), with 374 (+1), 318 (−1) and 227 (0). Using the bootstrap-sampling test described in (Koehn, 2004b), the improvement is statistically significant at $p < 0.05$. Though the improvement is modest, it is statistically significant and this positive result is important in view of the negative findings in (Carpuat and Wu, 2005) that WSD does not help MT. Furthermore, note that Hiero+WSD has higher *n*-gram precisions than Hiero.

7 Analysis

Ideally, the WSD system should be suggesting high-quality translations which are frequently part of the reference sentences. To determine this, we note the set of grammar rules used in the best derivation for translating each test sentence. From the rules of each test sentence, we tabulated the set of translations proposed by the WSD system and check whether they are found in the associated reference sentences.

On the entire set of NIST MT 2003 evaluation test sentences, an average of 10.36 translations proposed

No. of words in WSD translations	All test sentences		+1 from Collins sign-test	
	No. of WSD translations used	% match reference	No. of WSD translations used	% match reference
1	7087	77.31	3078	77.68
2	1930	66.11	861	64.92
3	371	43.13	171	48.54
4	124	26.61	52	28.85

Table 3: Number of WSD translations used and proportion that matches against respective reference sentences. WSD translations longer than 4 words are very sparse (less than 10 occurrences) and thus they are not shown.

by the WSD system were used for each sentence. When limited to the set of 374 sentences which were judged by the Collins sign-test to have better translations from Hiero+WSD than from Hiero, a higher number (11.14) of proposed translations were used on average. Further, for the entire set of test sentences, 73.01% of the proposed translations are found in the reference sentences. This increased to a proportion of 73.22% when limited to the set of 374 sentences. These figures show that having more, and higher-quality proposed translations contributed to the set of 374 sentences being better translations than their respective original translations from Hiero. Table 3 gives a detailed breakdown of these figures according to the number of words in each proposed translation. For instance, over all the test sentences, the WSD module gave 7087 translations of single-word length, and 77.31% of these translations match their respective reference sentences. We note that although the proportion of matching 2-word translations is slightly lower for the set of 374 sentences, the proportion increases for translations having more words.

After the experiments in Section 6 were completed, we visually inspected the translation output of Hiero and Hiero+WSD to categorize the ways in which integrating WSD contributes to better translations. The first way in which WSD helps is when it enables the integrated Hiero+WSD system to output extra appropriate English words. For example, the translations for the Chinese sentence "...或 其他 「恶劣 行为」， 将 无法 取得 更 多 援助 或 其他 让步 。" are as follows.

- Hiero: ... *or other bad behavior ", will be more aid and other concessions.*
- Hiero+WSD: ... *or other bad behavior ", will be unable to obtain more aid and other concessions.*

Here, the Chinese words "无法 取得" are not translated by Hiero at all. By providing the correct translation of "*unable to obtain*" for "无法 取得", the translation output of Hiero+WSD is more complete.

A second way in which WSD helps is by correcting a previously incorrect translation. For example, for the Chinese sentence "..., 在 全 国 各族 人 民， ...", the WSD system helps to correct Hiero's original translation by providing the correct translation of "*all ethnic groups*" for the Chinese phrase "各 族":

- Hiero: ..., *and people of all nationalities across the country, ...*
- Hiero+WSD: ..., *and people of all ethnic groups across the country, ...*

We also looked at the set of 318 sentences that were judged by the Collins sign-test to be worse translations. We found that in some situations, Hiero+WSD has provided extra appropriate English words, but those particular words are not used in the reference sentences. An interesting example is the translation of the Chinese sentence "澳洲 外长 指 北韩 行为 恶劣 将 无法 取得 更 多 援助".

- Hiero: *Australian foreign minister said that North Korea bad behavior will be more aid*
- Hiero+WSD: *Australian foreign minister said that North Korea bad behavior will be unable to obtain more aid*

This is similar to the example mentioned earlier. In this case however, those extra English words provided by Hiero+WSD, though appropriate, do not

result in more *n*-gram matches as the reference sentences used phrases such as "*will not gain*", "*will not get*", etc. Since the BLEU metric is precision based, the longer sentence translation by Hiero+WSD gets a lower BLEU score instead.

8 Conclusion

We have shown that WSD improves the translation performance of a state-of-the-art hierarchical phrase-based statistical MT system and this improvement is statistically significant. We have also demonstrated one way to integrate a WSD system into an MT system without introducing any rules that compete against existing rules, and where the feature-weight tuning and decoding place the WSD system on an equal footing with the other model components. For future work, an immediate step would be for the WSD classifier to provide translations for longer Chinese phrases. Also, different alternatives could be tried to match the translations provided by the WSD classifier against the chunks of rules. Finally, besides our proposed approach of integrating WSD into statistical MT via the introduction of two new features, we could explore other alternative ways of integration.

Acknowledgements

Yee Seng Chan is supported by a Singapore Millennium Foundation Scholarship (ref no. SMF-2004-1076). David Chiang was partially supported under the GALE program of the Defense Advanced Research Projects Agency, contract HR0011-06-C-0022.

References

C. Cabezas and P. Resnik. 2005. Using WSD techniques for lexical selection in statistical machine translation. Technical report, University of Maryland.

M. Carpuat and D. Wu. 2005. Word sense disambiguation vs. statistical machine translation. In *Proc. of ACL05*, pages 387–394.

M. Carpuat, W. Su, and D. Wu. 2004. Augmenting ensemble classification for word sense disambiguation with a kernel PCA model. In *Proc. of SENSEVAL-3*, pages 88–92.

C. C. Chang and C. J. Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. of ACL05*, pages 263–270.

D. Chiang. 2007. Hierarchical phrase-based translation. *To appear in Computational Linguistics*, 33(2).

M. Collins, P. Koehn, and I. Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proc. of ACL05*, pages 531–540.

U. Germann. 2003. Greedy decoding for statistical machine translation in almost linear time. In *Proc. of HLT-NAACL03*, pages 72–79.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proc. of HLT-NAACL03*, pages 48–54.

P. Koehn. 2003. *Noun Phrase Translation*. Ph.D. thesis, University of Southern California.

P. Koehn. 2004a. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proc. of AMTA04*, pages 115–124.

P. Koehn. 2004b. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP04*, pages 388–395.

Y. K. Lee and H. T. Ng. 2002. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proc. of EMNLP02*, pages 41–48.

P. M. II Lewis and R. E. Stearns. 1968. Syntax-directed transduction. *Journal of the ACM*, 15(3):465–488.

D. Marcu and W. Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proc. of EMNLP02*, pages 133–139.

F. J. Och and H. Ney. 2000. Improved statistical alignment models. In *Proc. of ACL00*, pages 440–447.

F. J. Och and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of ACL02*, pages 295–302.

F. J. Och and H. Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

F. J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL03*, pages 160–167.

K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proc. of ACL02*, pages 311–318.

A. Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proc. of ICSLP02*, pages 901–904.

D. Vickrey, L. Biewald, M. Teyssier, and D. Koller. 2005. Word-sense disambiguation for machine translation. In *Proc. of EMNLP05*, pages 771–778.

D. Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *Proc. of ACL96*, pages 152–158.

Learning Expressive Models for Word Sense Disambiguation

Lucia Specia
NILC/ICMC
University of São Paulo
Caixa Postal 668, 13560-970
São Carlos, SP, Brazil
lspecia@icmc.usp.br

Mark Stevenson
Department of Computer Science
University of Sheffield
Regent Court, 211 Portobello St.
Sheffield, S1 4DP, UK
marks@dcs.shef.ac.uk

Maria das Graças V. Nunes
NILC/ICMC
University of São Paulo
Caixa Postal 668, 13560-970
São Carlos, SP, Brazil
gracan@icmc.usp.br

Abstract

We present a novel approach to the word sense disambiguation problem which makes use of corpus-based evidence combined with background knowledge. Employing an inductive logic programming algorithm, the approach generates expressive disambiguation rules which exploit several knowledge sources and can also model relations between them. The approach is evaluated in two tasks: identification of the correct translation for a set of highly ambiguous verbs in English-Portuguese translation and disambiguation of verbs from the Senseval-3 lexical sample task. The average accuracy obtained for the multilingual task outperforms the other machine learning techniques investigated. In the monolingual task, the approach performs as well as the state-of-the-art systems which reported results for the same set of verbs.

1 Introduction

Word Sense Disambiguation (WSD) is concerned with the identification of the meaning of ambiguous words in context. For example, among the possible senses of the verb "run" are "to move fast by using one's feet" and "to direct or control". WSD can be useful for many applications, including information retrieval, information extraction and machine translation. Sense ambiguity has been recognized as one of the most important obstacles to successful language understanding since the early 1960's and many techniques have been proposed to solve the problem. Recent approaches focus on the use of various lexical resources and corpus-based techniques in order to avoid the substantial effort required to codify linguistic knowledge. These approaches have shown good results; particularly those using supervised learning (see Mihalcea et al., 2004 for an overview of state-of-the-art systems). However, current approaches rely on limited knowledge representation and modeling techniques: traditional machine learning algorithms and attribute-value vectors to represent disambiguation instances. This has made it difficult to exploit deep knowledge sources in the generation of the disambiguation models, that is, knowledge that goes beyond simple features extracted directly from the corpus, like bags-of-words and collocations, or provided by shallow natural language tools like part-of-speech taggers.

In this paper we present a novel approach for WSD that follows a hybrid strategy, i.e. combines knowledge and corpus-based evidence, and employs a first-order formalism to allow the representation of deep knowledge about disambiguation examples together with a powerful modeling technique to induce theories based on the examples and background knowledge. This is achieved using Inductive Logic Programming (ILP) (Muggleton, 1991), which has not yet been applied to WSD.

Our hypothesis is that by using a very expressive representation formalism, a range of (shallow and deep) knowledge sources and ILP as learning technique, it is possible to generate models that, when compared to models produced by machine learning algorithms conventionally applied to

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 41–48,
Prague, Czech Republic, June 2007.

WSD, are both more accurate for fine-grained distinctions, and "interesting", from a knowledge acquisition point of view (i.e., convey potentially new knowledge that can be easily interpreted by humans).

WSD systems have generally been more successful in the disambiguation of nouns than other grammatical categories (Mihalcea et al., 2004). A common approach to the disambiguation of nouns has been to consider a wide context around the ambiguous word and treat it as a bag of words or limited set of collocates. However, disambiguation of verbs generally benefits from more specific knowledge sources, such as the verb's relation to other items in the sentence (for example, by analysing the semantic type of its subject and object). Consequently, we believe that the disambiguation of verbs is task to which ILP is particularly well-suited. Therefore, this paper focuses on the disambiguation of verbs, which is an interesting task since much of the previous work on WSD has concentrated on the disambiguation of nouns.

WSD is usually approached as an independent task, however, it has been argued that different applications may have specific requirements (Resnik and Yarowsky, 1997). For example, in machine translation, WSD, or *translation disambiguation*, is responsible for identifying the correct *translation* for an ambiguous source word. There is not always a direct relation between the possible senses for a word in a (monolingual) lexicon and its translations to a particular language, so this represents a different task to WSD against a (monolingual) lexicon (Hutchins and Somers, 1992). Although it has been argued that WSD does not yield better translation quality than a machine translation system alone, it has been recently shown that a WSD module that is developed following specific multilingual requirements can significantly improve the performance of a machine translation system (Carpuat et al., 2006).

This paper focuses on the application of our approach to the translation of verbs in English to Portuguese translation, specifically for a set of 10 mainly light and highly ambiguous verbs. We also experiment with a monolingual task by using the verbs from Senseval-3 lexical sample task. We explore knowledge from 12 syntactic, semantic and pragmatic sources. In principle, the proposed approach could also be applied to any lexical disambiguation task by customizing the sense repository and knowledge sources.

In the remainder of this paper we first present related approaches to WSD and discuss their limitations (Section 2). We then describe some basic concepts on ILP and our application of this technique to WSD (Section 3). Finally, we described our experiments and their results (Section 4).

2 Related Work

WSD approaches can be classified as (a) knowledge-based approaches, which make use of linguistic knowledge, manually coded or extracted from lexical resources (Agirre and Rigau, 1996; Lesk 1986); (b) corpus-based approaches, which make use of shallow knowledge automatically acquired from corpus and statistical or machine learning algorithms to induce disambiguation models (Yarowsky, 1995; Schütze 1998); and (c) hybrid approaches, which mix characteristics from the two other approaches to automatically acquire disambiguation models from corpus supported by linguistic knowledge (Ng and Lee 1996; Stevenson and Wilks, 2001).

Hybrid approaches can combine advantages from both strategies, potentially yielding accurate and comprehensive systems, particularly when deep knowledge is explored. Linguistic knowledge is available in electronic resources suitable for practical use, such as WordNet (Fellbaum, 1998), dictionaries and parsers. However, the use of this information has been hampered by the limitations of the modeling techniques that have been explored so far: using deep sources of domain knowledge is beyond the capabilities of such techniques, which are in general based on attribute-value vector representations.

Attribute-value vectors consist of a set of attributes intended to represent properties of the examples. Each attribute has a type (its name) and a single value for a given example. Therefore, attribute-value vectors have the same expressiveness as propositional formalisms, that is, they only allow the representation of atomic propositions and constants. These are the representations used by most of the machine learning algorithms conventionally employed to WSD, for example Naïve Bayes and decision-trees. First-order logic, a more expressive formalism which is employed by ILP, allows the representation of variables and n-ary predicates, i.e., relational knowledge.

In the hybrid approaches that have been explored so far, deep knowledge, like selectional preferences, is either pre-processed into a vector representation to accommodate machine learning algorithms, or used in previous steps to filter out possible senses e.g. (Stevenson and Wilks, 2001). This may cause information to be lost and, in addition, deep knowledge sources cannot interact in the learning process. As a consequence, the models produced reflect only the shallow knowledge that is provided to the learning algorithm.

Another limitation of attribute-value vectors is the need for a unique representation for all the examples: one attribute is created for every knowledge feature and the same structure is used to characterize all the examples. This usually results in a very sparse representation of the data, given that values for certain features will not be available for many examples. The problem of data sparseness increases as more knowledge is exploited and this can cause problems for the machine learning algorithms.

A final disadvantage of attribute-value vectors is that equivalent features may have to be bounded to distinct identifiers. An example of this occurs when the syntactic relations between words in a sentence are represented by attributes for each possible relation, sentences in which there is more than one instantiation for a particular grammatical role cannot be easily represented. For example, the sentence "John and Anna gave Mary a present." contains a coordinate subject and, since each feature requires a unique identifier, two are required ($subj_1$-$verb_1$, $subj_2$-$verb_1$). These will be treated as two independent pieces of knowledge by the learning algorithm.

First-order formalisms allow a generic predicate to be created for every possible syntactic role, relating two or more elements. For example *has_subject(verb, subject)*, which could then have two instantiations: *has_subject(give, john)* and *has_subject(give, anna)*. Since each example is represented independently from the others, the data sparseness problem is minimized. Therefore, ILP seems to provide the most general-purpose framework for dealing with such data: it does not suffer from the limitations mentioned above since there are explicit provisions made for the inclusion of background knowledge of any form, and the representation language is powerful enough to capture contextual relationships.

3 A hybrid relational approach to WSD

In what follows we provide an introduction to ILP and then outline how it is applied to WSD by presenting the sample corpus and knowledge sources used in our experiments.

3.1 Inductive Logic Programming

Inductive Logic Programming (Muggleton, 1991) employs techniques from Machine Learning and Logic Programming to build first-order theories from examples and background knowledge, which are also represented by first-order clauses. It allows the efficient representation of substantial knowledge about the problem, which is used during the learning process, and produces disambiguation models that can make use of this knowledge. The general approach underlying ILP can be outlined as follows:

Given:

- a set of positive and negative examples $\boldsymbol{E} = \boldsymbol{E}^+ \cup \boldsymbol{E}^-$

- a predicate $\boldsymbol{p}$ specifying the target relation to be learned

- knowledge $\boldsymbol{K}$ of the domain, described according to a language $\boldsymbol{L_k}$, which specifies which predicates $\boldsymbol{q_i}$ can be part of the definition of $\boldsymbol{p}$.

The goal is: to induce a hypothesis (or theory) $\boldsymbol{h}$ for $\boldsymbol{p}$, with relation to $\boldsymbol{E}$ and $\boldsymbol{K}$, which covers most of the $\boldsymbol{E}^+$, without covering the $\boldsymbol{E}^-$, i.e., $\boldsymbol{K} \wedge \boldsymbol{h} \models \boldsymbol{E}^+$ and $\boldsymbol{K} \wedge \boldsymbol{h} \not\models \boldsymbol{E}^-$.

We use the Aleph ILP system (Srinivasan, 2000), which provides a complete inference engine and can be customized in various ways. The default inference engine induces a theory iteratively using the following steps:

1. One instance is randomly selected to be generalized.

2. A more specific clause (the bottom clause) is built using inverse entailment (Muggleton, 1995), generally consisting of the representation of all the knowledge about that example.

3. A clause that is more generic than the bottom clause is searched for using a given search (e.g., best-first) and evaluation strategy (e.g., number of positive examples covered).

4. The best clause is added to the theory and the examples covered by that clause are removed from the sample set. Stop if there are more no examples in the training set, otherwise return to step 1.

3.2 Sample data

This approach was evaluated using two scenarios: (1) an English-Portuguese multilingual setting addressing 10 very frequent and problematic verbs selected in a previous study (Specia et. al., 2005); and (2) an English setting consisting of 32 verbs from Senseval-3 lexical sample task (Mihalcea et. al. 2004).

For the first scenario a corpus containing 500 sentences for each of the 10 verbs was constructed. The text was randomly selected from corpora of different domains and genres, including literary fiction, Bible, computer science dissertation abstracts, operational system user manuals, newspapers and European Parliament proceedings. This corpus was automatically annotated with the translation of the verb using a tagging system based on parallel corpus, statistical information and translation dictionaries (Specia et al., 2005), followed by a manual revision. For each verb, the sense repository was defined as the set of all the possible translations of that verb in the corpus. 80% of the corpus was randomly selected and used for training, with the remainder retained for testing. The 10 verbs, number of possible translations and the percentage of sentences for each verb which use the most frequent translation are shown in Table 1.

For the monolingual scenario, we use the sense tagged corpus and sense repositories provided for verbs in Senseval-3. There are 32 verbs with between 40 and 398 examples each. The number of senses varies between 3 and 10 and the average percentage of examples with the majority (most frequent) sense is 55%.

Verb	# Translations	Most frequent translation - %
ask	7	53
come	29	36
get	41	13
give	22	72
go	30	53
live	8	66
look	12	41
make	21	70
take	32	25
tell	8	66

Table 1. Verbs and possible senses in our corpus

Both corpora were lemmatized and part-of-speech (POS) tagged using Minipar (Lin, 1993) and Mxpost (Ratnaparkhi, 1996), respectivelly. Additionally, proper nouns identified by the tagger were replaced by a single identifier (*proper_noun*) and pronouns replaced by identifiers representing classes of pronouns (*relative_pronoun*, etc.).

3.3 Knowledge sources

We now describe the background knowledge sources used by the learning algorithm, having as an example sentence (1), in which the word "coming" is the target verb being disambiguated.

(1) "If there is such a thing as reincarnation, I would not mind **coming** back as a squirrel".

KS_1. Bag-of-words consisting of 5 words to the right and left of the verb (excluding stop words), represented using definitions of the form *has_bag*(*snt, word*):

has_bag(snt_1, *mind*).
has_bag(snt_1, *not*). …

KS_2. Frequent bigrams consisting of pairs of adjacent words in a sentence (other than the target verb) which occur more than 10 times in the corpus, represented by *has_bigram(snt, $word_1$, $word_2$)*:

has_bigram(snt_1, back, as).
has_bigram(snt_1, such, a). …

KS_3. Narrow context containing 5 content words to the right and left of the verb, identified using POS tags, represented by *has_narrow*(*snt, word_position, word*):

has_narrow(snt_1, 1st_word_left, mind).
has_narrow(snt_1, 1st_word_right, back). …

KS_4. POS tags of 5 words to the right and left of the verb, represented by *has_pos*(*snt, word_position, pos*):

has pos(snt_1, 1st_word_left, nn).
has pos(snt_1, 1^{st}_word_right, rb). …

KS_5. 11 collocations of the verb: 1st preposition to the right, 1st and 2nd words to the left and right, 1st noun, 1st adjective, and 1st verb to the left and right. These are represented using definitions of the form *has_collocation*(*snt, type, collocation*):

has_collocation(snt_1, *1st_prep_right, back*).
has_collocation(snt_1, *1st_noun_left, mind*).…

KS_6. Subject and object of the verb obtained using Minipar and represented by *has_rel*(*snt, type, word*):

has_rel(snt$_1$, subject, i).
has_rel(snt$_1$, object, nil). …

KS_7. Grammatical relations not including the target verb also identified using Minipar. The relations (verb-subject, verb-object, verb-modifier, subject-modifier, and object-modifier) occurring more than 10 times in the corpus are represented by *has_related_pair(snt, word$_1$, word$_2$)*:

has_related_pair(snt$_1$, there, be). …

KS_8. The sense with the highest count of overlapping words in its dictionary definition and in the sentence containing the target verb (excluding stop words) (Lesk, 1986), represented by *has_overlapping*(*sentence, translation*):

has_overlapping(snt$_1$, voltar).

KS_9. Selectional restrictions of the verbs defined using LDOCE (Procter, 1978). WordNet is used when the restrictions imposed by the verb are not part of the description of its arguments, but can be satisfied by synonyms or hyperonyms of those arguments. A hierarchy of feature types is used to account for restrictions established by the verb that are more generic than the features describing its arguments in the sentence. This information is represented by definitions of the form *satisfy_restriction*(*snt, rest_subject, rest_object*):

satisfy_restriction(*snt$_1$,* [*human*], *nil*).
satisfy_restriction(*snt$_1$,* [*animal, human*], *nil*).

KS_1-KS_9 can be applied to both multilingual and monolingual disambiguation tasks. The following knowledge sources were specifically designed for multilingual applications:

KS_{10}. Phrasal verbs in the sentence identified using a list extracted from various dictionaries. (This information was not used in the monolingual task because phrasal constructions are not considered verb senses in Senseval data.) These are represented by definitions of the form *has_expression*(*snt, verbal_expression*):

has_expression(snt$_1$, "come back").

KS_{11}. Five words to the right and left of the target verb in the Portuguese translation. This could be obtained using a machine translation system that would first translate the non-ambiguous words in the sentence. In our experiments it was extracted using a parallel corpus and represented using definitions of the form *has_bag_trns*(*snt, portuguese_word*):

has_bag_trns(snt$_1$, coelho).
has_bag_trns(snt$_1$, reincarnação). …

KS_{12}. Narrow context consisting of 5 collocations of the verb in the Portuguese translation, which take into account the positions of the words, represented by *has_narrow_trns*(*snt, word_position, portuguese_word*):

has_narrow_trns(*snt*$_1$, *1st_word_right, como*).
has_narrow_trns(*snt*$_1$, *2nd_word_right, um*). …

In addition to background knowledge, the system learns from a set of examples. Since all knowledge about them is expressed as background knowledge, their representation is very simple, containing only the sentence identifier and the sense of the verb in that sentence, i.e. *sense(snt, sense)*:

sense(snt$_1$,voltar).
sense(snt$_2$,ir). …

Based on the examples, background knowledge and a series of settings specifying the predicate to be learned (i.e., the heads of the rules), the predicates that can be in the conditional part of the rules, how the arguments can be shared among different predicates and several other parameters, the inference engine produces a set of symbolic rules. Figure 1 shows examples of the rules induced for the verb "to come" in the multilingual task.

Rule_1. sense(A, voltar) :-
  has_collocation(A, 1st_prep_right, back).
Rule_2. sense(A, chegar) :-
  has_rel(A, subj, B), has_bigram(A, today, B),
  has_bag_trans(A, hoje).
Rule_3. sense(A, chegar) :-
  satisfy_restriction(A, [animal, human], [concrete]);
  has_expression(A, 'come at').
Rule_4. sense(A, vir) :-
  satisfy_restriction(A, [animate], nil);
  (has_rel(A, subj, B),
  (has_pos(A, B, nnp); has_pos(A, B, prp))).

Figure 1. Examples of rules produced for the verb "come" in the multilingual task

Models learned with ILP are symbolic and can be easily interpreted. Additionally, innovative knowledge about the problem can emerge from the rules learned by the system. Although some rules simply test shallow features such as collocates, others pose conditions on sets of knowledge sources, including relational sources, and allow non-instantiated arguments to be shared amongst them by means of variables. For example, in Figure 1, **Rule_1** states that the translation of the verb in a sentence *A* will be "voltar" (*return*) if the first preposition to the right of the verb in that sentence is "back". **Rule_2** states that the translation of the verb will be "chegar" (*arrive*) if it has a certain subject *B*, which occurs frequently with the word "today" as a bigram, and if the partially translated sentence contains the word "hoje" (the translation of "today"). **Rule_3** says that the translation of the verb will be "chegar" (*reach*) if the subject of the verb has the features "animal" or "human" and the object has the feature "concrete", or if the verb occurs in the expression "come at". **Rule_4** states that the translation of the verb will be "vir" (*move toward*) if the subject of the verb has the feature "animate" and there is no object, or if the verb has a subject *B* that is a proper noun (*nnp*) or a personal pronoun (*prp*).

4 Experiments and results

To assess the performance of the approach the model produced for each verb was tested on the corresponding set of test cases by applying the rules in a decision-list like approach, i.e., retaining the order in which they were produced and backing off to the most frequent sense in the training set to classify cases that were not covered by any of the rules. All the knowledge sources were made available to be used by the inference engine, since previous experiments showed that they are all relevant (Specia, 2006). In what follows we present the results and discuss each task.

4.1 Multilingual task

Table 2 shows the accuracies (in terms of percentage of corpus instances which were correctly disambiguated) obtained by the Aleph models. Results are compared against the accuracy that would be obtained by using the most frequent translation in the training set to classify all the examples of the test set (in the column labeled "Majority sense"). For comparison, we ran experiments with three learning algorithms frequently used for WSD, which rely on knowledge represented as attribute-value vectors: C4.5 (decision-trees), Naive Bayes and Support Vector Machine (SVM)[1]. In order to represent all knowledge sources in attribute-value vectors, KS_2, KS_7, KS_9 and KS_{10} had to be pre-processed to be transformed into binary attributes. For example, in the case of selectional restrictions (KS_9), one attribute was created for each possible sense of the verb and a true/false value was assigned to it depending on whether the arguments of the verb satisfied any restrictions referring to that sense. Results for each of these algorithms are also shown in Table 2.

As we can see in Table 2, the accuracy of the ILP approach is considerably better than the most frequent sense baseline and also outperforms the other learning algorithms. This improvement is statistically significant (paired t-test; $p < 0.05$). As expected, accuracy is generally higher for verbs with fewer possible translations.

The models produced by Aleph for all the verbs are reasonably compact, containing 50 to 96 rules. In those models the various knowledge sources appear in different rules and all are used. This demonstrates that they are all useful for the disambiguation of verbs.

Verb	Majority sense	C4.5	Naïve Bayes	SVM	Aleph
ask	0.68	0.68	0.82	0.88	0.92
come	0.46	0.57	0.61	0.68	0.73
get	0.03	0.25	0.46	0.47	0.49
give	0.72	0.71	0.74	0.74	0.74
go	0.49	0.61	0.66	0.66	0.66
live	0.71	0.72	0.64	0.73	0.87
look	0.48	0.69	0.81	0.83	0.93
make	0.64	0.62	0.60	0.64	0.68
take	0.14	0.41	0.50	0.51	0.59
tell	0.65	0.67	0.66	0.68	0.82
Average	**0.50**	**0.59**	**0.65**	**0.68**	**0.74**

Table 2. Accuracies obtained by Aleph and other learning algorithms in the multilingual task

These results are very positive, particularly if we consider the characteristics of the multilingual scenario: (1) the verbs addressed are highly ambiguous; (2) the corpus was automatically tagged and thus distinct synonym translations were sometimes

[1] The implementations provided by Weka were used. Weka is available from http://www.cs.waikato.ac.nz/ml/weka/

used to annotate different examples (these count as different senses for the inference engine); and (3) certain translations occur very infrequently (just 1 or 2 examples in the whole corpus). It is likely that a less strict evaluation regime, such as one which takes account of synonym translations, would result in higher accuracies.

It is worth noticing that we experimented with a few relevant parameters for both Aleph and the other learning algorithms. Values that yielded the best average predictive accuracy in the training sets were assumed to be optimal and used to evaluate the test sets.

4.2 Monolingual task

Table 3 shows the average accuracy obtained by Aleph in the monolingual task (Senseval-3 verbs with fine-grained sense distinctions and using the evaluation system provided by Senseval). It also shows the average accuracy of the most frequent sense and accuracies reported on the same set of verbs by the best systems submitted by the sites which participated in this task. Syntalex-3 (Mohammad and Pedersen, 2004) is based on an ensemble of bagged decision trees with narrow context part-of-speech features and bigrams. CLaC1 (Lamjiri et al., 2004) uses a Naive Bayes algorithm with a dynamically adjusted context window around the target word. Finally, MC-WSD (Ciaramita and Johnson, 2004) is a multi-class averaged perceptron classifier using syntactic and narrow context features, with one component trained on the data provided by Senseval and other trained on WordNet glosses.

System	% Average accuracy
Majority sense	0.56
Syntalex-3	0.67
CLaC1	0.67
MC-WSD	0.72
Aleph	0.72

Table 3. Accuracies obtained by Aleph and other approaches in the monolingual task

As we can see in Table 3, results are very encouraging: even without being particularly customized for this monolingual task, the ILP approach significantly outperforms the majority sense baseline and performs as well as the state-of-the-art system reporting results for the same set of verbs. As with the multilingual task, the models produced contain a small number of rules (from 6, for verbs with a few examples, to 88) and all knowledge sources are used across different rules and verbs.

In general, results from both multilingual and monolingual tasks demonstrate that the hypothesis put forward in Section 1, that ILP's ability to generate expressive rules which combine and integrate a wide range of knowledge sources is beneficial for WSD systems, is correct.

5 Conclusion

We have introduced a new hybrid approach to WSD which uses ILP to combine deep and shallow knowledge sources. ILP induces expressive disambiguation models which include relations between knowledge sources. It is an interesting approach to learning which has been considered promising for several applications in natural language processing and has been explored for a few of them, namely POS-tagging, grammar acquisition and semantic parsing (Cussens et al., 1997; Mooney, 1997). This paper has demonstrated that ILP also yields good results for WSD, in particular for the disambiguation of verbs.

We plan to further evaluate our approach for other sets of words, including other parts-of-speech to allow further comparisons with other approaches. For example, Dang and Palmer (2005) also use a rich set of features with a traditional learning algorithm (maximum entropy). Currently, we are evaluating the role of the WSD models for the 10 verbs of the multilingual task in an English-Portuguese statistical machine translation system.

References

Eneko Agirre and German Rigau. 1996. Word Sense Disambiguation using Conceptual Density. *Proceedings of the 15th Conference on Computational Linguistics (COLING-96)*. Copenhagen, pages 16-22.

Marine Carpuat, Yihai Shen, Xiaofeng Yu, and Dekai WU. 2006. Toward Integrating Word Sense and Entity Disambiguation into Statistical Machine Translation. *Proceedings of the Third International Workshop on Spoken Language Translation*,. Kyoto, pages 37-44.

Massimiliano Ciaramita and Mark Johnson. 2004. Multi-component Word Sense Disambiguation. *Proceedings of Senseval-3: 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, pages 97-100.

James Cussens, David Page, Stephen Muggleton, and Ashwin Srinivasan. 1997. Using Inductive Logic Programming for Natural Language Processing. *Workshop Notes on Empirical Learning of Natural Language Tasks*, Prague, pages 25-34.

Hoa T. Dang and Martha Palmer. 2005. The Role of Semantic Roles in Disambiguating Verb Senses. *Proceedings of the 43rd Meeting of the Association for Computational Linguistics (ACL-05)*, Ann Arbor, pages 42–49.

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

W. John Hutchins and Harold L. Somers. 1992. *An Introduction to Machine Translation*. Academic Press, Great Britain.

Abolfazl K. Lamjiri, Osama El Demerdash, Leila Kosseim. 2004. Simple features for statistical Word Sense Disambiguation. *Proceedings of Senseval-3: 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, pages 133-136.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. *ACM SIGDOC Conference*, Toronto, pages 24-26.

Dekang Lin. 1993. Principle based parsing without overgeneration. *Proceedings of the 31st Meeting of the Association for Computational Linguistics (ACL-93)*, Columbus, pages 112-120.

Rada Mihalcea, Timothy Chklovski and Adam Kilgariff. 2004. The Senseval-3 English Lexical Sample Task. *Proceedings of Senseval-3: 3rd International Workshop on the Evaluation of Systems for Semantic Analysis of Text*, Barcelona, pages 25-28.

Saif Mohammad and Ted Pedersen. 2004. Complementarity of Lexical and Simple Syntactic Features: The SyntaLex Approach to Senseval-3. *Proceedings of Senseval-3: 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, pages 159-162.

Raymond J. Mooney. 1997. Inductive Logic Programming for Natural Language Processing. *Proceedings of the 6th International Workshop on ILP*, LNAI 1314, Stockolm, pages 3-24.

Stephen Muggleton. 1991. Inductive Logic Programming. *New Generation Computing*, 8(4):295-318.

Stephen Muggleton. 1995. Inverse Entailment and Progol. *New Generation Computing*, 13:245-286.

Hwee T. Ng and Hian B. Lee. 1996. Integrating multiple knowledge sources to disambiguate word sense: an exemplar-based approach. *Proceedings of the 34th Meeting of the Association for Computational Linguistics (ACL-96)*, Santa Cruz, CA, pages 40-47.

Paul Procter (editor). 1978. *Longman Dictionary of Contemporary English*. Longman Group, Essex.

Adwait Ratnaparkhi. 1996. A Maximum Entropy Part-Of-Speech Tagger. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, New Jersey, pages 133-142.

Phillip Resnik and David Yarowsky. 1997. A Perspective on Word Sense Disambiguation Methods and their Evaluating. *Proceedings of the ACL-SIGLEX Workshop Tagging Texts with Lexical Semantics: Why, What and How*?, Washington.

Hinrich Schütze. 1998. Automatic Word Sense Discrimination. *Computational Linguistics*, 24(1): 97-123.

Lucia Specia, Maria G.V. Nunes, and Mark Stevenson. 2005. Exploiting Parallel Texts to Produce a Multilingual Sense Tagged Corpus for Word Sense Disambiguation. *Proceedings of the Conference on Recent Advances on Natural Language Processing (RANLP-2005)*, Borovets, pages 525-531.

Lucia Specia. 2006. A Hybrid Relational Approach for WSD - First Results. *Proceedings of the COLING/ACL 06 Student Research Workshop*, Sydney, pages 55-60.

Ashwin Srinivasan. 2000. The Aleph Manual. *Technical Report*. Computing Laboratory, Oxford University.

Mark Stevenson and Yorick Wilks. 2001. The Interaction of Knowledge Sources for Word Sense Disambiguation. *Computational Linguistics*, 27(3):321-349.

Yorick Wilks and Mark Stevenson. 1998. The Grammar of Sense: Using Part-of-speech Tags as a First Step in Semantic Disambiguation. *Journal of Natural Language Engineering*, 4(1):1-9

David Yarowsky. 1995. Unsupervised Word-Sense Disambiguation Rivaling Supervised Methods. *Proceedings of the 33rd Meeting of the Association for Computational Linguistics (ACL-05)*, Cambridge, MA, pages 189-196.

Domain Adaptation with Active Learning for Word Sense Disambiguation

Yee Seng Chan and **Hwee Tou Ng**
Department of Computer Science
National University of Singapore
3 Science Drive 2, Singapore 117543
{chanys, nght}@comp.nus.edu.sg

Abstract

When a word sense disambiguation (WSD) system is trained on one domain but applied to a different domain, a drop in accuracy is frequently observed. This highlights the importance of domain adaptation for word sense disambiguation. In this paper, we first show that an active learning approach can be successfully used to perform domain adaptation of WSD systems. Then, by using the predominant sense predicted by expectation-maximization (EM) and adopting a count-merging technique, we improve the effectiveness of the original adaptation process achieved by the basic active learning approach.

1 Introduction

In natural language, a word often assumes different meanings, and the task of determining the correct meaning, or sense, of a word in different contexts is known as word sense disambiguation (WSD). To date, the best performing systems in WSD use a corpus-based, supervised learning approach. With this approach, one would need to collect a text corpus, in which each ambiguous word occurrence is first tagged with its correct sense to serve as training data.

The reliance of supervised WSD systems on annotated corpus raises the important issue of domain dependence. To investigate this, Escudero et al. (2000) and Martinez and Agirre (2000) conducted experiments using the DSO corpus, which contains sentences from two different corpora, namely Brown Corpus (BC) and Wall Street Journal (WSJ). They found that training a WSD system on one part (BC or WSJ) of the DSO corpus, and applying it to the other, can result in an accuracy drop of more than 10%, highlighting the need to perform domain adaptation of WSD systems to new domains. Escudero et al. (2000) pointed out that one of the reasons for the drop in accuracy is the difference in sense priors (i.e., the proportions of the different senses of a word) between BC and WSJ. When the authors assumed they knew the sense priors of each word in BC and WSJ, and adjusted these two datasets such that the proportions of the different senses of each word were the same between BC and WSJ, accuracy improved by 9%.

In this paper, we explore domain adaptation of WSD systems, by adding training examples from the new domain as additional training data to a WSD system. To reduce the effort required to adapt a WSD system to a new domain, we employ an active learning strategy (Lewis and Gale, 1994) to select examples to annotate from the new domain of interest. To our knowledge, our work is the first to use active learning for domain adaptation for WSD. A similar work is the recent research by Chen et al. (2006), where active learning was used successfully to reduce the annotation effort for WSD of 5 English verbs using *coarse-grained* evaluation. In that work, the authors only used active learning to reduce the annotation effort and did not deal with the porting of a WSD system to a new domain.

Domain adaptation is necessary when the training and target domains are different. In this paper,

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 49–56,
Prague, Czech Republic, June 2007.

we perform domain adaptation for WSD of a set of nouns using *fine-grained* evaluation. The contribution of our work is not only in showing that active learning can be successfully employed to reduce the annotation effort required for domain adaptation in a *fine-grained* WSD setting. More importantly, our main focus and contribution is in showing how we can improve the effectiveness of a basic active learning approach when it is used for domain adaptation. In particular, we explore the issue of different sense priors across different domains. Using the sense priors estimated by expectation-maximization (EM), the predominant sense in the new domain is predicted. Using this predicted predominant sense and adopting a count-merging technique, we *improve* the effectiveness of the adaptation process.

In the next section, we discuss the choice of corpus and nouns used in our experiments. We then introduce active learning for domain adaptation, followed by count-merging. Next, we describe an EM-based algorithm to estimate the sense priors in the new domain. Performance of domain adaptation using active learning and count-merging is then presented. Next, we show that by using the predominant sense of the target domain as predicted by the EM-based algorithm, we improve the effectiveness of the adaptation process. Our empirical results show that for the set of nouns which have different predominant senses between the training and target domains, we are able to reduce the annotation effort by 71%.

2 Experimental Setting

In this section, we discuss the motivations for choosing the particular corpus and the set of nouns to conduct our domain adaptation experiments.

2.1 Choice of Corpus

The DSO corpus (Ng and Lee, 1996) contains 192,800 annotated examples for 121 nouns and 70 verbs, drawn from BC and WSJ. While the BC is built as a balanced corpus, containing texts in various categories such as religion, politics, humanities, fiction, etc, the WSJ corpus consists primarily of business and financial news. Exploiting the difference in coverage between these two corpora, Escudero et al. (2000) separated the DSO corpus into its BC and WSJ parts to investigate the domain dependence of several WSD algorithms. Following the setup of (Escudero et al., 2000), we similarly made use of the DSO corpus to perform our experiments on domain adaptation.

Among the few currently available manually sense-annotated corpora for WSD, the SEMCOR (SC) corpus (Miller et al., 1994) is the most widely used. SEMCOR is a subset of BC which is sense-annotated. Since BC is a balanced corpus, and since performing adaptation from a general corpus to a more specific corpus is a natural scenario, we focus on adapting a WSD system trained on BC to WSJ in this paper. Henceforth, out-of-domain data will refer to BC examples, and in-domain data will refer to WSJ examples.

2.2 Choice of Nouns

The WordNet Domains resource (Magnini and Cavaglia, 2000) assigns domain labels to synsets in WordNet. Since the focus of the WSJ corpus is on business and financial news, we can make use of WordNet Domains to select the set of nouns having at least one synset labeled with a business or finance related domain label. This is similar to the approach taken in (Koeling et al., 2005) where they focus on determining the predominant sense of words in corpora drawn from finance versus sports domains.[1] Hence, we select the subset of DSO nouns that have at least one synset labeled with any of these domain labels: *commerce*, *enterprise*, *money*, *finance*, *banking*, and *economy*. This gives a set of 21 nouns: *book*, *business*, *center*, *community*, *condition*, *field*, *figure*, *house*, *interest*, *land*, *line*, *money*, *need*, *number*, *order*, *part*, *power*, *society*, *term*, *use*, *value*.[2]

For each noun, all the BC examples are used as out-of-domain training data. One-third of the WSJ examples for each noun are set aside as evaluation

[1] Note however that the coverage of the WordNet Domains resource is not comprehensive, as about 31% of the synsets are simply labeled with "factotum", indicating that the synset does not belong to a specific domain.

[2] 25 nouns have at least one synset labeled with the listed domain labels. In our experiments, 4 out of these 25 nouns have an accuracy of more than 90% before adaptation (i.e., training on just the BC examples) and accuracy improvement is less than 1% after all the available WSJ adaptation examples are added as additional training data. To obtain a clearer picture of the adaptation process, we discard these 4 nouns, leaving a set of 21 nouns.

Dataset	No. of senses		MFS acc. (%)	No. of training examples	No. of adaptation examples
	BC	WSJ			
21 nouns	6.7	6.8	61.1	310	406
9 nouns	7.9	8.6	65.8	276	416

Table 1: The average number of senses in BC and WSJ, average MFS accuracy, average number of BC training, and WSJ adaptation examples per noun.

data, and the rest of the WSJ examples are designated as in-domain adaptation data. The row *21 nouns* in Table 1 shows some information about these 21 nouns. For instance, these nouns have an average of 6.7 senses in BC and 6.8 senses in WSJ. This is slightly higher than the 5.8 senses per verb in (Chen et al., 2006), where the experiments were conducted using coarse-grained evaluation. Assuming we have access to an "oracle" which determines the predominant sense, or most frequent sense (MFS), of each noun in our WSJ test data perfectly, and we assign this most frequent sense to each noun in the test data, we will have achieved an accuracy of 61.1% as shown in the column *MFS accuracy* of Table 1. Finally, we note that we have an average of 310 BC training examples and 406 WSJ adaptation examples per noun.

3 Active Learning

For our experiments, we use naive Bayes as the learning algorithm. The knowledge sources we use include parts-of-speech, local collocations, and surrounding words. These knowledge sources were effectively used to build a state-of-the-art WSD program in one of our prior work (Lee and Ng, 2002). In performing WSD with a naive Bayes classifier, the sense s assigned to an example with features $f_1, \ldots, f_n$ is chosen so as to maximize:

$$p(s) \prod_{j=1}^{n} p(f_j|s)$$

In our domain adaptation study, we start with a WSD system built using training examples drawn from BC. We then investigate the utility of adding additional in-domain training data from WSJ. In the baseline approach, the additional WSJ examples are randomly selected. With active learning (Lewis and Gale, 1994), we use *uncertainty sampling* as shown in Figure 1. In each iteration, we train a WSD system on the available training data and apply it on the WSJ adaptation examples. Among these WSJ examples, the example predicted with the lowest confidence is selected and removed from the adaptation data. The correct label is then supplied for this example and it is added to the training data.

$D_T \leftarrow$ the set of BC training examples
$D_A \leftarrow$ the set of untagged WSJ adaptation examples
$\Gamma \leftarrow$ WSD system trained on D_T
repeat
  $p_{min} \leftarrow \infty$
  for each $d \in D_A$ do
    $\widehat{s} \leftarrow$ word sense prediction for d using Γ
    $p \leftarrow$ confidence of prediction $\widehat{s}$
    if $p < p_{min}$ then
      $p_{min} \leftarrow p$, $d_{min} \leftarrow d$
    end
  end
  $D_A \leftarrow D_A - d_{min}$
  provide correct sense s for d_{min} and add d_{min} to D_T
  $\Gamma \leftarrow$ WSD system trained on new D_T
end

Figure 1: Active learning

Note that in the experiments reported in this paper, all the adaptation examples are already pre-annotated before the experiments start, since all the WSJ adaptation examples come from the DSO corpus which have already been sense-annotated. Hence, the annotation of an example needed during each adaptation iteration is simulated by performing a lookup without any manual annotation.

4 Count-merging

We also employ a technique known as *count-merging* in our domain adaptation study. Count-merging assigns different weights to different examples to better reflect their relative importance. Roark and Bacchiani (2003) showed that weighted count-merging is a special case of maximum a posteriori (MAP) estimation, and successfully used it for probabilistic context-free grammar domain adaptation (Roark and Bacchiani, 2003) and language model adaptation (Bacchiani and Roark, 2003).

Count-merging can be regarded as scaling of counts obtained from different data sets. We let $\widetilde{c}$ denote the counts from out-of-domain training data, $\bar{c}$ denote the counts from in-domain adaptation data, and $\widehat{p}$ denote the probability estimate by

count-merging. We can scale the out-of-domain and in-domain counts with different factors, or just use a single weight parameter β:

$$\widehat{p}(f_j|s_i) = \frac{\widetilde{c}(f_j, s_i) + \beta\bar{c}(f_j, s_i)}{\widetilde{c}(s_i) + \beta\bar{c}(s_i)} \quad (1)$$

Similarly,

$$\widehat{p}(s_i) = \frac{\widetilde{c}(s_i) + \beta\bar{c}(s_i)}{\widetilde{c} + \beta\bar{c}} \quad (2)$$

Obtaining an optimum value for β is not the focus of this work. Instead, we are interested to see if assigning a higher weight to the in-domain WSJ adaptation examples, as compared to the out-of-domain BC examples, will improve the adaptation process. Hence, we just use a β value of 3 in our experiments involving count-merging.

5 Estimating Sense Priors

In this section, we describe an EM-based algorithm that was introduced by Saerens et al. (2002), which can be used to estimate the sense priors, or a priori probabilities of the different senses in a new dataset. We have recently shown that this algorithm is effective in estimating the sense priors of a set of nouns (Chan and Ng, 2005).

Most of this section is based on (Saerens et al., 2002). Assume we have a set of labeled data D_L with *n* classes and a set of *N* independent instances $(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ from a new data set. The likelihood of these *N* instances can be defined as:

$$\begin{aligned} L(\mathbf{x}_1, \ldots, \mathbf{x}_N) &= \prod_{k=1}^{N} p(\mathbf{x}_k) \\ &= \prod_{k=1}^{N} \left[\sum_{i=1}^{n} p(\mathbf{x}_k, \omega_i) \right] \\ &= \prod_{k=1}^{N} \left[\sum_{i=1}^{n} p(\mathbf{x}_k|\omega_i) p(\omega_i) \right] \quad (3) \end{aligned}$$

Assuming the within-class densities $p(\mathbf{x}_k|\omega_i)$, i.e., the probabilities of observing $\mathbf{x}_k$ given the class ω_i, do not change from the training set D_L to the new data set, we can define: $p(\mathbf{x}_k|\omega_i) = p_L(\mathbf{x}_k|\omega_i)$. To determine the a priori probability estimates $\widehat{p}(\omega_i)$ of the new data set that will maximize the likelihood of (3) with respect to $p(\omega_i)$, we can apply the iterative procedure of the EM algorithm. In effect, through maximizing the likelihood of (3), we obtain the a priori probability estimates as a by-product.

Let us now define some notations. When we apply a classifier trained on D_L on an instance $\mathbf{x}_k$ drawn from the new data set D_U, we get $\widehat{p}_L(\omega_i|\mathbf{x}_k)$, which we define as the probability of instance $\mathbf{x}_k$ being classified as class ω_i by the classifier trained on D_L. Further, let us define $\widehat{p}_L(\omega_i)$ as the a priori probability of class ω_i in D_L. This can be estimated by the class frequency of ω_i in D_L. We also define $\widehat{p}^{(s)}(\omega_i)$ and $\widehat{p}^{(s)}(\omega_i|\mathbf{x}_k)$ as estimates of the new a priori and a posteriori probabilities at step *s* of the iterative EM procedure. Assuming we initialize $\widehat{p}^{(0)}(\omega_i) = \widehat{p}_L(\omega_i)$, then for each instance $\mathbf{x}_k$ in D_U and each class ω_i, the EM algorithm provides the following iterative steps:

$$\widehat{p}^{(s)}(\omega_i|\mathbf{x}_k) = \frac{\widehat{p}_L(\omega_i|\mathbf{x}_k)\frac{\widehat{p}^{(s)}(\omega_i)}{\widehat{p}_L(\omega_i)}}{\sum_{j=1}^{n} \widehat{p}_L(\omega_j|\mathbf{x}_k)\frac{\widehat{p}^{(s)}(\omega_j)}{\widehat{p}_L(\omega_j)}} \quad (4)$$

$$\widehat{p}^{(s+1)}(\omega_i) = \frac{1}{N}\sum_{k=1}^{N} \widehat{p}^{(s)}(\omega_i|\mathbf{x}_k) \quad (5)$$

where Equation (4) represents the expectation E-step, Equation (5) represents the maximization M-step, and *N* represents the number of instances in D_U. Note that the probabilities $\widehat{p}_L(\omega_i|\mathbf{x}_k)$ and $\widehat{p}_L(\omega_i)$ in Equation (4) will stay the same throughout the iterations for each particular instance $\mathbf{x}_k$ and class ω_i. The new a posteriori probabilities $\widehat{p}^{(s)}(\omega_i|\mathbf{x}_k)$ at step *s* in Equation (4) are simply the a posteriori probabilities in the conditions of the labeled data, $\widehat{p}_L(\omega_i|\mathbf{x}_k)$, weighted by the ratio of the new priors $\widehat{p}^{(s)}(\omega_i)$ to the old priors $\widehat{p}_L(\omega_i)$. The denominator in Equation (4) is simply a normalizing factor.

The a posteriori $\widehat{p}^{(s)}(\omega_i|\mathbf{x}_k)$ and a priori probabilities $\widehat{p}^{(s)}(\omega_i)$ are re-estimated sequentially during each iteration *s* for each new instance $\mathbf{x}_k$ and each class ω_i, until the convergence of the estimated probabilities $\widehat{p}^{(s)}(\omega_i)$, which will be our estimated sense priors. This iterative procedure will increase the likelihood of (3) at each step.

6 Experimental Results

For each adaptation experiment, we start off with a classifier built from an initial training set consisting

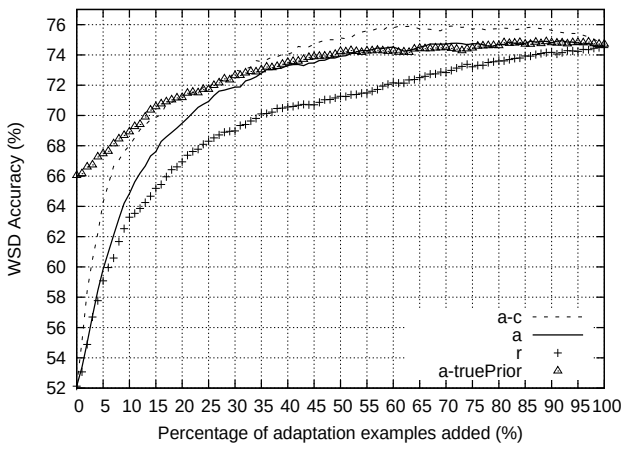


Figure 2: Adaptation process for all 21 nouns.

of the BC training examples. At each adaptation iteration, WSJ adaptation examples are selected *one at a time* and added to the training set. The adaptation process continues until all the adaptation examples are added. Classification accuracies averaged over 3 random trials on the WSJ test examples at each iteration are calculated. Since the number of WSJ adaptation examples differs for each of the 21 nouns, the learning curves we will show in the various figures are plotted in terms of different percentage of adaptation examples added, varying from 0 to 100 percent in steps of 1 percent. To obtain these curves, we first calculate for each noun, the WSD accuracy when different percentages of adaptation examples are added. Then, for each percentage, we calculate the macro-average WSD accuracy over all the nouns to obtain a single learning curve representing all the nouns.

6.1 Utility of Active Learning and Count-merging

In Figure 2, the curve *r* represents the adaptation process of the baseline approach, where additional WSJ examples are randomly selected during each adaptation iteration. The adaptation process using active learning is represented by the curve *a*, while applying count-merging with active learning is represented by the curve *a-c*. Note that random selection *r* achieves its highest WSD accuracy after *all* the adaptation examples are added. To reach the same accuracy, the *a* approach requires the addition of only 57% of adaptation examples. The *a-c* approach is even more effective and requires only 42% of adaptation examples. This demonstrates the effectiveness of count-merging in further reducing the annotation effort, when compared to using only active learning. To reach the MFS accuracy of 61.1% as shown earlier in Table 1, *a-c* requires just 4% of the adaptation examples.

To determine the utility of the out-of-domain BC examples, we have also conducted three active learning runs using only WSJ adaptation examples. Using 10%, 20%, and 30% of WSJ adaptation examples to build a classifier, the accuracy of these runs is lower than the active learning *a* curve and paired t-tests show that the difference is statistically significant at the level of significance 0.01.

6.2 Using Sense Priors Information

As mentioned in section 1, research in (Escudero et al., 2000) noted an improvement in accuracy when they adjusted the BC and WSJ datasets such that the proportions of the different senses of each word were the same between BC and WSJ. We can similarly choose BC examples such that the sense priors in the BC training data adhere to the sense priors in the WSJ evaluation data. To gauge the effectiveness of this approach, we first assume that we know the *true* sense priors of each noun in the WSJ evaluation data. We then gather BC training examples for a noun to adhere as much as possible to the sense priors in WSJ. Assume sense s_i is the predominant sense in the WSJ evaluation data, s_i has a sense prior of p_i in the WSJ data and has n_i BC training examples. Taking n_i examples to represent a sense prior of p_i, we proportionally determine the number of BC examples to gather for other senses s according to their respective sense priors in WSJ. If there are insufficient training examples in BC for some sense s, whatever available examples of s are used.

This approach gives an average of 195 BC training examples for the 21 nouns. With this new set of training examples, we perform adaptation using active learning and obtain the *a-truePrior* curve in Figure 2. The *a-truePrior* curve shows that by ensuring that the sense priors in the BC training data adhere as much as possible to the sense priors in the WSJ data, we start off with a higher WSD accuracy. However, the performance is no different from the *a*

curve after 35% of adaptation examples are added. A possible reason might be that by strictly adhering to the sense priors in the WSJ data, we have removed too many BC training examples, from an average of 310 examples per noun as shown in Table 1, to an average of 195 examples.

6.3 Using Predominant Sense Information

Research by McCarthy et al. (2004) and Koeling et al. (2005) pointed out that a change of predominant sense is often indicative of a change in domain. For example, the predominant sense of the noun *interest* in the BC part of the DSO corpus has the meaning "a sense of concern with and curiosity about someone or something". In the WSJ part of the DSO corpus, the noun *interest* has a different predominant sense with the meaning "a fixed charge for borrowing money", which is reflective of the business and finance focus of the WSJ corpus.

Instead of restricting the BC training data to adhere strictly to the sense priors in WSJ, another alternative is just to ensure that the predominant sense in BC is the same as that of WSJ. Out of the 21 nouns, 12 nouns have the same predominant sense in both BC and WSJ. The remaining 9 nouns that have different predominant senses in the BC and WSJ data are: *center, field, figure, interest, line, need, order, term, value*. The row *9 nouns* in Table 1 gives some information for this set of 9 nouns. To gauge the utility of this approach, we conduct experiments on these nouns by first assuming that we know the *true* predominant sense in the WSJ data. Assume that the WSJ predominant sense of a noun is s_i and s_i has n_i examples in the BC data. We then gather BC examples for a noun to adhere to this WSJ predominant sense, by gathering only up to n_i BC examples for each sense of this noun. This approach gives an average of 190 BC examples for the 9 nouns. This is higher than an average of 83 BC examples for these 9 nouns if BC examples are selected to follow the sense priors of WSJ evaluation data as described in the last subsection 6.2.

For these 9 nouns, the average KL-divergence between the sense priors of the original BC data and WSJ evaluation data is 0.81. This drops to 0.51 after ensuring that the predominant sense in BC is the same as that of WSJ, confirming that the sense priors in the newly gathered BC data more closely follow

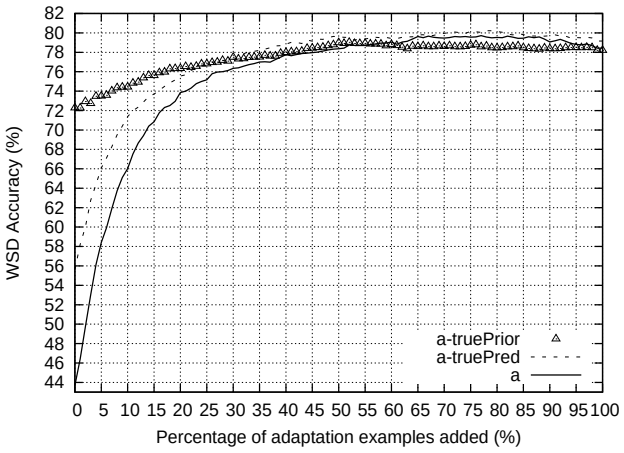


Figure 3: Using true predominant sense for the 9 nouns.

the sense priors in WSJ. Using this new set of training examples, we perform domain adaptation using active learning to obtain the curve *a-truePred* in Figure 3. For comparison, we also plot the curves *a* and *a-truePrior* for this set of 9 nouns in Figure 3. Results in Figure 3 show that *a-truePred* starts off at a higher accuracy and performs consistently better than the *a* curve. In contrast, though *a-truePrior* starts at a high accuracy, its performance is lower than *a-truePred* and *a* after 50% of adaptation examples are added. The approach represented by *a-truePred* is a compromise between ensuring that the sense priors in the training data follow as closely as possible the sense priors in the evaluation data, while retaining enough training examples. These results highlight the importance of striking a balance between these two goals.

In (McCarthy et al., 2004), a method was presented to determine the predominant sense of a word in a corpus. However, in (Chan and Ng, 2005), we showed that in a supervised setting where one has access to some annotated training data, the EM-based method in section 5 estimates the sense priors more effectively than the method described in (McCarthy et al., 2004). Hence, we use the EM-based algorithm to estimate the sense priors in the WSJ evaluation data for each of the 21 nouns. The sense with the highest estimated sense prior is taken as the predominant sense of the noun.

For the set of 12 nouns where the predominant

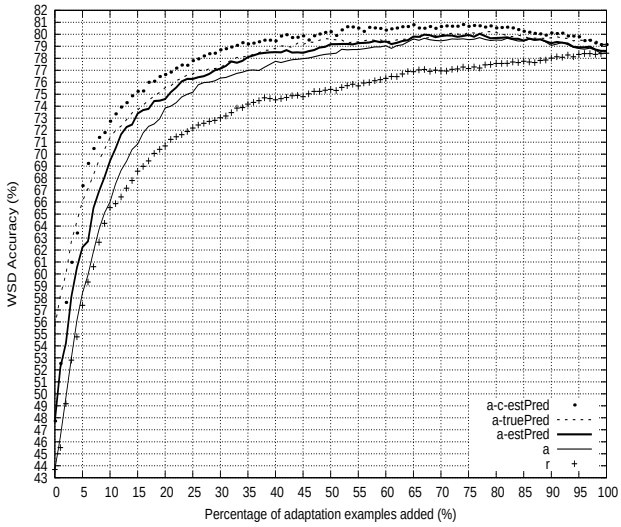


Figure 4: Using estimated predominant sense for the 9 nouns.

Accuracy	% adaptation examples needed			
	r	a	a-estPred	a-c-estPred
50%: 61.1	8	7 (0.88)	5 (0.63)	4 (0.50)
60%: 64.5	10	9 (0.90)	7 (0.70)	5 (0.50)
70%: 68.0	15	12 (0.80)	9 (0.60)	6 (0.40)
80%: 71.5	23	16 (0.70)	12 (0.52)	9 (0.39)
90%: 74.9	46	24 (0.52)	21 (0.46)	15 (0.33)
100%: 78.4	100	51 (0.51)	38 (0.38)	29 (0.29)

Table 2: Annotation savings and percentage of adaptation examples needed to reach various accuracies.

sense remains unchanged between BC and WSJ, the EM-based algorithm is able to predict that the predominant sense remains unchanged for *all* 12 nouns. Hence, we will focus on the 9 nouns which have different predominant senses between BC and WSJ for our remaining adaptation experiments. For these 9 nouns, the EM-based algorithm correctly predicts the WSJ predominant sense for 6 nouns. Hence, the algorithm is able to predict the correct predominant sense for 18 out of 21 nouns overall, representing an accuracy of 86%.

Figure 4 plots the curve *a-estPred*, which is similar to *a-truePred*, except that the predominant sense is now estimated by the EM-based algorithm. Employing count-merging with *a-estPred* produces the curve *a-c-estPred*. For comparison, the curves *r*, *a*, and *a-truePred* are also plotted. The results show that *a-estPred* performs consistently better than *a*, and *a-c-estPred* in turn performs better than *a-estPred*. Hence, employing the predicted predominant sense and count-merging, we further improve the effectiveness of the active learning-based adaptation process.

With reference to Figure 4, the WSD accuracies of the *r* and *a* curves before and after adaptation are 43.7% and 78.4% respectively. Starting from the mid-point 61.1% accuracy, which represents a 50% accuracy increase from 43.7%, we show in Table 2 the percentage of adaptation examples required by the various approaches to reach certain levels of WSD accuracies. For instance, to reach the final accuracy of 78.4%, *r*, *a*, *a-estPred*, and *a-c-estPred* require the addition of 100%, 51%, 38%, and 29% adaptation examples respectively. The numbers in brackets give the ratio of adaptation examples needed by *a*, *a-estPred*, and *a-c-estPred* versus random selection *r*. For instance, to reach a WSD accuracy of 78.4%, *a-c-estPred* needs only 29% adaptation examples, representing a ratio of 0.29 and an annotation saving of 71%. Note that this represents a more effective adaptation process than the basic active learning *a* approach, which requires 51% adaptation examples. Hence, besides showing that active learning can be used to reduce the annotation effort required for domain adaptation, we have further improved the effectiveness of the adaptation process by using the predicted predominant sense of the new domain and adopting the count-merging technique.

7 Related Work

In applying active learning for domain adaptation, Zhang et al. (2003) presented work on sentence boundary detection using generalized Winnow, while Tur et al. (2004) performed language model adaptation of automatic speech recognition systems. In both papers, out-of-domain and in-domain data were simply mixed together without MAP estimation such as count-merging. For WSD, Fujii et al. (1998) used selective sampling for a Japanese language WSD system, Chen et al. (2006) used active learning for 5 verbs using coarse-grained evaluation, and H. T. Dang (2004) employed active learning for another set of 5 verbs. However, their work only investigated the use of active learning to reduce the annotation effort necessary for WSD, but

did not deal with the porting of a WSD system to a different domain. Escudero et al. (2000) used the DSO corpus to highlight the importance of the issue of domain dependence of WSD systems, but did not propose methods such as active learning or count-merging to address the specific problem of how to perform domain adaptation for WSD.

8 Conclusion

Domain adaptation is important to ensure the general applicability of WSD systems across different domains. In this paper, we have shown that active learning is effective in reducing the annotation effort required in porting a WSD system to a new domain. Also, we have successfully used an EM-based algorithm to detect a change in predominant sense between the training and new domain. With this information on the predominant sense of the new domain and incorporating count-merging, we have shown that we are able to improve the effectiveness of the original adaptation process achieved by the basic active learning approach.

Acknowledgement

Yee Seng Chan is supported by a Singapore Millennium Foundation Scholarship (ref no. SMF-2004-1076).

References

M. Bacchiani and B. Roark. 2003. Unsupervised language model adaptation. In *Proc. of IEEE ICASSP03*.

Y. S. Chan and H. T. Ng. 2005. Word sense disambiguation with distribution estimation. In *Proc. of IJCAI05*.

J. Chen, A. Schein, L. Ungar, and M. Palmer. 2006. An empirical study of the behavior of active learning for word sense disambiguation. In *Proc. of HLT/NAACL06*.

H. T. Dang. 2004. *Investigations into the Role of Lexical Semantics in Word Sense Disambiguation*. PhD dissertation, University of Pennsylvania.

G. Escudero, L. Marquez, and G. Rigau. 2000. An empirical study of the domain dependence of supervised word sense disambiguation systems. In *Proc. of EMNLP/VLC00*.

A. Fujii, K. Inui, T. Tokunaga, and H. Tanaka. 1998. Selective sampling for example-based word sense disambiguation. *Computational Linguistics*, 24(4).

R. Koeling, D. McCarthy, and J. Carroll. 2005. Domain-specific sense distributions and predominant sense acquisition. In *Proc. of Joint HLT-EMNLP05*.

Y. K. Lee and H. T. Ng. 2002. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proc. of EMNLP02*.

D. D. Lewis and W. A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proc. of SIGIR94*.

B. Magnini and G. Cavaglia. 2000. Integrating subject field codes into WordNet. In *Proc. of LREC-2000*.

D. Martinez and E. Agirre. 2000. One sense per collocation and genre/topic variations. In *Proc. of EMNLP/VLC00*.

D. McCarthy, R. Koeling, J. Weeds, and J. Carroll. 2004. Finding predominant word senses in untagged text. In *Proc. of ACL04*.

G. A. Miller, M. Chodorow, S. Landes, C. Leacock, and R. G. Thomas. 1994. Using a semantic concordance for sense identification. In *Proc. of HLT94 Workshop on Human Language Technology*.

H. T. Ng and H. B. Lee. 1996. Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In *Proc. of ACL96*.

B. Roark and M. Bacchiani. 2003. Supervised and unsupervised PCFG adaptation to novel domains. In *Proc. of HLT-NAACL03*.

M. Saerens, P. Latinne, and C. Decaestecker. 2002. Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Computation*, 14(1).

D. H. Tur, G. Tur, M. Rahim, and G. Riccardi. 2004. Unsupervised and active learning in automatic speech recognition for call classification. In *Proc. of IEEE ICASSP04*.

T. Zhang, F. Damerau, and D. Johnson. 2003. Updating an NLP system to fit new domains: an empirical study on the sentence segmentation problem. In *Proc. of CONLL03*.

Making Lexical Ontologies Functional and Context-Sensitive

Tony Veale
Computer Science and Informatics
University College Dublin
Ireland
tony.veale@ucd.ie

Yanfen Hao
Computer Science and Informatics
University College Dublin
Ireland
yanfen.hao@ucd.ie

Abstract

Human categorization is neither a binary nor a context-free process. Rather, some concepts are better examples of a category than others, while the criteria for category membership may be satisfied to different degrees by different concepts in different contexts. In light of these empirical facts, WordNet's static category structure appears both excessively rigid and unduly fragile for processing real texts. In this paper we describe a syntagmatic, corpus-based approach to redefining WordNet's categories in a functional, gradable and context-sensitive fashion. We describe how the diagnostic properties for these definitions are automatically acquired from the web, and how the increased flexibility in categorization that arises from these redefinitions offers a robust account of metaphor comprehension in the mold of Glucksberg's (2001) theory of category-inclusion. Furthermore, we demonstrate how this competence with figurative categorization can effectively be governed by automatically-generated ontological constraints, also acquired from the web.

1 Introduction

Linguistic variation across contexts is often symptomatic of ontological differences between contexts. These observable variations can serve as valuable clues not just to the specific senses of words in context (e.g., see Pustejovsky, Hanks and Rumshisky, 2004) but to the underlying ontological structure itself (see Cimiano, Hotho and Staab, 2005). The most revealing variations are syntagmatic in nature, which is to say, they look beyond individual word forms to larger patterns of contiguous usage (Hanks, 2004). In most contexts, the similarity between chocolate, say, and a narcotic like heroin will meagerly reflect the simple ontological fact that both are kinds of substances; certainly, taxonomic measures of similarity as discussed in Budanitsky and Hirst (2006) will capture little more than this commonality. However, in a context in which the addictive properties of chocolate are very salient (e.g., an online dieting forum), chocolate is more likely to be categorized as a drug and thus be considered more similar to heroin. Look, for instance, at the similar ways in which these words can be used: one can be ”chocolate-crazed” or ”chocolate-addicted” and suffer ”chocolate-induced” symptoms (e.g., each of these uses can be found in the pages of Wikipedia). In a context that gives rise to these expressions, it is unsurprising that chocolate should appear altogether more similar to a harmful narcotic.

In this paper we computationally model this idea that language use reflects category structure. As noted by De Leenheer and de Moor (2005), ontologies are lexical representations of concepts, so we can expect the effects of context on language use to closely reflect the effects of context on ontological structure. An understanding of the linguistic effects of context, as expressed through syntagmatic patterns of word usage, should lead therefore to the design of more flexible lexical ontologies that naturally adapt to their contexts of use. WordNet (Fell-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 57–64,
Prague, Czech Republic, June 2007.

baum, 1998) is just one such lexical ontology that can benefit greatly from the added flexibility that context-sensitivity can bring. Though comprehensive in scale and widely used, WordNet suffers from an obvious structural rigidity in which concepts are either entirely within a category or entirely outside a category: no gradation of category membership is allowed, and no contextual factors are brought to bear on criteria for membership. Thus, a gun is always a weapon in WordNet while an axe is never so, despite the uses (sporting or murderous) to which each can be put.

In section two we describe a computational framework for giving WordNet senses a functional, context-sensitive form. These functional forms simultaneously represent i) an intensional definition for each word sense; ii) a structured query capable of retrieving instances of the corresponding category from a context-specific corpus; and iii) a membership function that assigns gradated scores to these instances based on available syntagmatic evidence. In section three we describe how the knowledge required to automate this functional re-definition is acquired from the web and linked to WordNet. In section four we describe how these re-definitions can produce a robust model of metaphor, before we evaluate the descriptive sufficiency of this approach in section five, comparing it to the knowledge already available within WordNet. We conclude with some final remarks in section six.

2 Functional Context-Sensitive Categories

We take a wholly textual view of context and assume that a given context can be implicitly characterized by a representative text corpus. This corpus can be as large as a text archive or an encyclopedia (e.g., the complete text of Wikipedia), or as small as a single document, a sentence or even a single noun-phrase. For instance, the micro-context "alcoholic apple-juice" is enough to implicate the category Liquor, rather than Juice, as a semantic head, while "lovable snake" can be enough of a context to locally categorize Snake as a kind of Pet. There is a range of syntagmatic patterns that one can exploit to glean category insights from a text. For instance, the "X kills" pattern is enough to categorize X as a kind of Killer, "hunts X" is enough to categorize X as a kind of Prey, while "X-covered", "X-dipped" and "X-frosted" all indicate that X is a kind of Covering. Likewise, "army of X" suggests that a context views X as a kind of Soldier, while "barrage of X" suggests that X should be seen as a kind of Projectile.

We operationalize the collocation-type of adjective and noun via the function (*attr ADJ NOUN*), which returns a number in the range 0...1; this represents the extent to which ADJ is used to modify NOUN in the context-defining corpus. Dice's coefficient (e.g., see Cimiano *et al*., 2005) is used to implement this measure. A context-sensitive category membership function can be defined, as in that for Fundamentalist in Figure 1:

```
(define Fundamentalist.0 (arg0)
    (*  (max
            (%isa arg0 Person.0)
            (%isa arg0 Group.0))
        (min
            (max
                (attr political arg0)
                (attr religious arg0))
            (max
                (attr extreme arg0)
                (attr violent arg0)
                (attr radical arg0)))))
```

Figure 1. A functional re-definition of the category Fundamentalist.

The function of Figure 1 takes, as a single argument $\mathbf{arg}_0$, a putative member of the category Fundamentalist.0 (note how the sense tag, 0, is used to identify a specific WordNet sense of "fundamentalist"), and returns a membership score in the range 0...1 for this term. This score reflects the syntagmatic evidence for considering $\mathbf{arg}_0$ to be political or religious, as well as extreme or violent or radical. The function *(%isa* $\mathbf{arg}_0$ *CAT)* returns a value of 1.0 if some sense of $\mathbf{arg}_0$ is a descendent of CAT (here Person.0 or Group.0), otherwise 0. This safeguards ontological coherence and ensures that only kinds of people or groups can ever be considered as fundamentalists.

The example of Figure 1 is hand-crafted, but a functional form can be assigned automatically to many of the synsets in WordNet by heuristic means.

For instance, those of Figure 2 are automatically derived from WordNet's morpho-semantic links:

*(define Fraternity.0 (**arg**$_0$)*
((%sim **arg**$_0$ Fraternity.0)*
(max
*(attr fraternal **arg**$_0$)*
*(attr brotherly **arg**$_0$))))*

*(define Orgasm.0 (**arg**$_0$)*
((%sim **arg**$_0$ Orgasm.0)*
(max
*(attr climactic **arg**$_0$)*
*(attr orgasmic **arg**$_0$))))*

Figure 2. Exploiting the WordNet links between nouns and their adjectival forms.

The function *(%sim **arg**$_0$ CAT)* reflects the perceived similarity between the putative member arg_0 and a synset CAT in WordNet, using one of the standard formulations described in Budanitsky and Hirst (2006). Thus, any kind of group (e.g., a glee club, a Masonic lodge, or a barbershop quartet) described in a text as "fraternal" or "brotherly" (both occupy the same WordNet synset) can be considered a Fraternity to the corresponding degree, tempered by its *a priori* similarity to a Fraternity; likewise, any climactic event can be categorized as an Orgasm to a more or less degree.

Alternately, the function of Figure 3 is automatically obtained for the lexical concept Espresso by shallow parsing its WordNet gloss: "strong black coffee brewed by forcing steam under pressure through powdered coffee beans".

*(define Espresso.0 (**arg**$_0$)*
((%sim **arg**$_0$ Espresso.0)*
(min
*(attr strong **arg**$_0$)*
*(attr black **arg**$_0$))))*

Figure 3. A functional re-definition of the category Espresso based on its WordNet gloss.

It follows that any substance (e.g., oil or tea) described locally as "black" and "strong" with a non-zero taxonomic similarity to coffee can be considered a kind of Espresso.

Combining the contents of WordNet 1.6 and WordNet 2.1, 27,732 different glosses (shared by 51,035 unique word senses) can be shallow parsed to yield a definition of the kind shown in Figure 3. Of these, 4525 glosses yield two or more properties that can be given functional form via *attr*. However, one can question whether these features are sufficient, and more importantly, whether they are truly diagnostic of the categories they are used to define. In the next section we consider another source of diagnostic properties, explicit similes on the web, before, in section 5, comparing the quality of these properties to those available from WordNet.

3 Diagnostic Properties on the Web

We employ the *Google* search engine as a retrieval mechanism for acquiring the diagnostic properties of categories from the web, since the *Google* API and its support for the wildcard term * allows this process to be fully automated. The guiding intuition here is that looking for explicit similes of the form "X is as P as Y" is the surest way of finding the most salient properties of a term Y; with other syntagmatic patterns, such as adjective:noun collocations, one cannot be sure that the adjective is central to the noun.

Since we expect that explicit similes will tend to exploit properties that occupy an exemplary point on a scale, we first extract a list of antonymous adjectives, such as "hot" or "cold", from WordNet. For every adjective ADJ on this list, we send the query *"as ADJ as *"* to Google and scan the first 200 snippets returned to extract different noun values for the wildcard *. From each set of snippets we can also ascertain the relative frequencies of different noun values for ADJ. The complete set of nouns extracted in this way is then used to drive a second phase of the search, in which the query template *"as * as a NOUN"* is used to acquire similes that may have lain beyond the 200-snippet horizon of the original search, or that may hinge on adjectives not included on the original list. Together, both phases collect a wide-ranging series of core samples (of 200 hits each) from across the web, yielding a set of 74,704 simile instances (of 42,618 unique types) relating

3769 different adjectives to 9286 different nouns

3.1 Property Filtering

Unfortunately, many of these similes are not sufficiently well-formed to identify salient properties. In many cases, the noun value forms part of a larger noun phrase: it may be the modifier of a compound noun (as in "bread lover"), or the head of complex noun phrase (such as "gang of thieves" or "wound that refuses to heal"). In the former case, the compound is used if it corresponds to a compound term in WordNet and thus constitutes a single lexical unit; if not, or if the latter case, the simile is rejected. Other similes are simply too contextual or underspecified to function well in a null context, so if one must read the original document to make sense of the simile, it is rejected. More surprisingly, perhaps, a substantial number of the retrieved similes are ironic, in which the literal meaning of the simile is contrary to the meaning dictated by common sense. For instance, "as hairy as a bowling ball" (found once) is an ironic way of saying "as hairless as a bowling ball" (also found just once). Many ironies can only be recognized using world knowledge, such as "as sober as a Kennedy" and "as tanned as an Irishman".

Given the creativity involved in these constructions, one cannot imagine a reliable automatic filter to safely identify bona-fide similes. For this reason, the filtering task is performed by a human judge, who annotated 30,991 of these simile instances (for 12,259 unique adjective/noun pairings) as non-ironic and meaningful in a null context; these similes relate a set of 2635 adjectives to a set of 4061 different nouns. In addition, the judge also annotated 4685 simile instances (of 2798 types) as ironic; these similes relate a set of 936 adjectives to a set of 1417 nouns. Perhaps surprisingly, ironic pairings account for over 13% of all annotated simile instances and over 20% of all annotated types.

3.2 Linking to WordNet Senses

To create functional WordNet definitions from these adjective:noun pairings, we first need to identify the WordNet sense of each noun. For instance, "as stiff as a zombie" might refer either to a re-animated corpse or to an alcoholic cocktail (both are senses of "zombie" in WordNet, and drinks can be "stiff" too). Disambiguation is trivial for nouns with just a single sense in WordNet. For nouns with two or more fine-grained senses that are all taxonomically close, such as "gladiator" (two senses: a boxer and a combatant), we consider each sense to be a suitable target. In some cases, the WordNet gloss for as particular sense will literally mention the adjective of the simile, and so this sense is chosen. In all other cases, we employ a strategy of mutual disambiguation to relate the noun vehicle in each simile to a specific sense in WordNet. Two similes "as A as N_1" and "as A as N_2" are mutually disambiguating if N_1 and N_2 are synonyms in WordNet, or if some sense of N_1 is a hypernym or hyponym of some sense of N_2 in WordNet. For instance, the adjective "scary" is used to describe both the noun "rattler" and the noun "rattlesnake" in bona-fide (non-ironic) similes; since these nouns share a sense, we can assume that the intended sense of "rattler" is that of a dangerous snake rather than a child's toy. Similarly, the adjective "brittle" is used to describe both saltines and crackers, suggesting that it is the bread sense of "cracker" rather than the hacker, firework or hillbilly senses (all in WordNet) that is intended.

These heuristics allow us to automatically disambiguate 10,378 bona-fide simile types (85%), yielding a mapping of 2124 adjectives to 3778 different WordNet senses. Likewise, 77% (or 2164) of the simile types annotated as ironic are disambiguated automatically. A remarkable stability is observed in the alignment of noun vehicles to WordNet senses: 100% of the ironic vehicles always denote the same sense, no matter the adjective involved, while 96% of bona-fide vehicles always denote the same sense. This stability suggests two conclusions: the disambiguation process is consistent and accurate; but more intriguingly, only one coarse-grained sense of any word is likely to be sufficiently exemplary of some property to be useful in a simile.

4 From Similes to Category Functions

As noted in section 3, the filtered web data yields 12,259 bona-fide similes describing 4061 target nouns in terms of 2635 different adjectival properties. Word-sense disambiguation allows 3778 synsets in WordNet to be given a functional redefinition in terms of 2124 diagnostic properties, as

in the definition of Gladiator in Figure 4:

*(define Gladiator.0 (**arg**$_0$)*
((%isa **arg**$_0$ Person.0)*
((%sim **arg**$_0$ Gladiator.0)*
(combine
*(attr strong **arg**$_0$)*
*(attr violent **arg**$_0$)*
*(attr manly **arg**$_0$)))))*

Figure 4. A web-based definition of Gladiator.

Since we cannot ascertain from the web data which properties are necessary and which are collectively sufficient, we use the function *combine* to aggregate the available evidence. This function implements a naïve probabilistic *or*, in which each piece of syntagmatic evidence is naively assumed to be independent, as follows:

(combine $e_0\, e_1$) = e_0 + $e_1(1 - e_0)$
(combine $e_0\, e_{1 \ldots} e_n$) = (combine e_0 (combine $e_{1 \ldots} e_n$))

Thus, any combatant or competitor (such as a sportsman) that is described as *strong*, *violent* or *manly* in a corpus can be categorized as a Gladiator in that context; the more properties that hold, and the greater the degree to which they hold, the greater the membership score that is assigned.

The source of the hard taxonomic constraint *(%isa* arg_0 *Person.0)* is explained in the next section. For now, note how the use of *%sim* in the functions of Figures 2, 3 and 4 means that these membership functions readily admit both literal and metaphoric members. Since the line between literal and metaphoric uses of a category is often impossible to draw, the best one can do is to accept metaphor as a gradable phenomenon (see Hanks, 2006). The incorporation of taxonomic similarity via *%sim* ensures that literal members will tend to receive higher membership scores, and that the most tenuous metaphors will receive the lowest membership scores (close to 0.0).

4.1 Constrained Category Inclusion

Simile and metaphor involve quite different conceptual mechanisms. For instance, anything that is particularly strong or black might meaningfully be called ”as black as espresso” or ”as strong as espresso”, yet few such things can meaningfully be called just ”espresso”. While simile is a mechanism for highlighting inter-concept similarity, metaphor is at heart a mechanism of category inclusion (see Glucksberg, 2001). As the espresso example demonstrates, category inclusion is more than a matter of shared properties: humans have strong intuitions about the structure of categories and the extent to which they can be stretched to include new members. So while it is sensible to apply the category Espresso to other substances, preferably liquids, it seems nonsensical to apply the category to animals, artifacts, places and so on.

Much as the salient properties of categories can be acquired form the web (see section 3), so too can the intuitions governing inclusion amongst categories. For instance, an attested web-usage of the phrase ”Espresso-like CAT” tells us that sub-types of CAT are allowable targets of categorization by the category Espresso. Thus, since the query ”espresso-like substance” returns 3 hits via Google, types of substance (oil, etc.) can be described as Espresso if they are contextually strong and black. In contrast, the query ”espresso-like person” returns 0 hits, so no instance of person can be described as Espresso, no matter how black or how strong. While this is clearly a heuristic approach to a complex cognitive problem, it does allow us to tap into the tacit knowledge that humans employ in categorization. More generally, a concept X can be included in a category C if X exhibits salient properties of C and, for some hypernym H of X in WordNet, we can find an attested use of ”C-like H” on the web.

If we can pre-fetch all possible ”C-like H” from the web, this will allow comprehension to proceed without having to resort to web analysis in mid-categorization. While there are too many possible values of H to make full pre-fetching a practical reality, we *can* generalize the problem somewhat, by selecting a range of values for H from the middle-layer of WordNet, such as *Person*, *Substance*, *Animal*, *Tool*, *Plant*, *Structure*, *Event*, *Vehicle*, *Idea* and *Place*, and by pre-fetching the query ”C-like H” for all 4061 nouns collected in section 3, combined with this limited set of H values. For every noun in our database then, we precompile a vector of possible category inclusions.

For instance, ”lattice” yields the following vector:

{*structure(1620), substance(8), container(1), vehicle(1)*}

where numbers in parentheses indicate the web-frequency of the corresponding ”Lattice-like H” query. Thus, the category Lattice can be used to describe (and metaphorically include) other kinds of structure (like crystals), types of substance (e.g., crystalline substances), containers (like honeycombs) and even vehicles (e.g., those with many compartments). Likewise, the noun ”snake” yields the following vector of possibilities:

{*structure(125), animal(122), person(56), vehicle(17), tool(9)*}

(note, the frequency for ”person” includes the frequency for ”man” and ”woman”). The category Snake can also be used to describe and include structures (like tunnels), other animals (like eels), people (e.g., the dishonest variety), vehicles (e.g., articulated trucks, trains) and tools (e.g., hoses). The noun ”gladiator” yields a vector of just one element, {*person(1)*}, from which the simple constraint *(%isa* arg_0 *Person.0)* in Figure 4 is derived. In contrast, ”snake” is now given the definition of Figure 5:

```
(define Snake.0 (arg0)
      (*  (max
            (%isa arg0 Structure.0)
            (%isa arg0 Animal.0)
            (%isa arg0 Person.0)
            (%isa arg0 Vehicle.0))
          (* (%sim arg0 Snake.0)
             (combine
                 (attr cunning arg0)
                 (attr slippery arg0)
                 (attr flexible arg0)
                 (attr slim arg0)
                 (attr sinuous arg0)
                 (attr crooked arg0)
                 (attr deadly arg0)
                 (attr poised arg0)))))
```

Figure 5. A membership function for Snake using web-derived category-inclusion constraints.

Glucksberg (2001) notes that the same category, used figuratively, can exhibit different qualities in different metaphors. For instance, Snake might describe a kind of crooked person in one metaphor, a poised killer in another metaphor, and a kind of flexible tool in yet another. The use of *combine* in Figure 5 means that a single category definition can give rise to each of these perspectives in the appropriate contexts. We therefore do not need a different category definition for each metaphoric use of Snake.

To illustrate the high-level workings of category-inclusion, Table 1 generalizes over the set of 3778 disambiguated nouns from section 3 to estimate the propensity for one semantic category, like Person, to include members of another category, like Animal, in X-like Y constructs.

X-like Y	*P*	*A*	*Sub*	*T*	*Str*
(P)erson	.66	.05	.03	.04	.09
(A)nimal	.36	.27	.04	.05	.15
(Sub)stance	.14	.03	.37	.05	.32
(T)ool	.08	.03	.07	.22	.34
(Str)ucture	.04	.03	.03	.03	.43

Table 1. The Likelihood of a category X accommodating a category Y.

Table 1 reveals that 36% of ”ANIMAL-like” patterns on the web describe a kind of Person, while only 5% of ”PERSON-like” patterns on the web describe a kind of Animal. Category inclusion appears here to be a conservative mechanism, with like describing like in most cases; thus, types of Person are most often used to describe other kinds of Person (comprising 66% of ”PERSON-like” patterns), types of substance to describe other substances, and so on. The clear exception is Animal, with ”ANIMAL-like” phrases more often used to describe people (36%) than other kinds of animal (27%). The anthropomorphic uses of this category demonstrate the importance of folk-knowledge in figurative categorization, of the kind one is more likely to find in real text, and on the web (as in section 3), rather than in resources like WordNet.

5 Empirical Evaluation

The simile gathering process of section 3, abetted by Google's practice of ranking pages according to popularity, should reveal the most frequently-used comparative nouns, and thus, the most useful categories to capture in a general-purpose ontology like WordNet. But the descriptive sufficiency of these categories is not guaranteed unless the defining properties ascribed to each can be shown to be collectively rich enough, and individually salient enough, to predict how each category is perceived and applied by a language user.

If similes are indeed a good basis for mining the most salient and diagnostic properties of categories, we should expect the set of properties for each category to accurately predict how the category is perceived as a whole. For instance, humans – unlike computers – do not generally adopt a dispassionate view of ideas, but rather tend to associate certain positive or negative feelings, or affective values, with particular ideas. Unsavoury activities, people and substances generally possess a negative affect, while pleasant activities and people possess a positive affect. Whissell (1989) reduces the notion of affect to a single numeric dimension, to produce a *dictionary of affect* that associates a numeric value in the range 1.0 (most unpleasant) to 3.0 (most pleasant) with over 8000 words in a range of syntactic categories (including adjectives, verbs and nouns). So to the extent that the adjectival properties yielded by processing similes paint an accurate picture of each category / noun-sense, we should be able to predict the affective rating of each vehicle via a weighted average of the affective ratings of the adjectival properties ascribed to these nouns (i.e., where the affect rating of each adjective contributes to the estimated rating of a noun in proportion to its frequency of co-occurrence with that noun in our simile data). More specifically, we should expect that ratings estimated via these simile-derived properties should correlate well with the independent ratings contained in Whissell's dictionary.

To determine whether similes do offer the clearest perspective on a category's most salient properties, we calculate and compare this correlation using the following data sets:

A. Adjectives derived from annotated bona-fide (non-ironic) similes only.

B. Adjectives derived from all annotated similes (both ironic and non-ironic).

C. Adjectives derived from ironic similes only.

D. All adjectives used to modify a given noun in a large corpus. We use over 2-gigabytes of text from the online encyclopaedia Wikipedia as our corpus.

E. The set of 63,935 unique property-of-noun pairings extracted via shallow-parsing from WordNet glosses in section 2, e.g., *strong* and *black* for Espresso.

Predictions of affective rating were made from each of these data sources and then correlated with the ratings reported in Whissell's dictionary of affect using a two-tailed Pearson test ($p < 0.01$). As expected, property sets derived from bona-fide similes only (A) yielded the best correlation (+0.514) while properties derived from ironic similes only (C) yielded the worst (-0.243); a middling correlation coefficient of 0.347 was found for all similes together, demonstrating the fact that bona-fide similes outnumber ironic similes by a ratio of 4 to 1. A weaker correlation of 0.15 was found using the corpus-derived adjectival modifiers for each noun (D); while this data provides quite large property sets for each noun, these properties merely reflect potential rather than intrinsic properties of each noun and so do not reveal what is most diagnostic about a category. More surprisingly, property sets derived from WordNet glosses (E) are also poorly predictive, yielding a correlation with Whissell's affect ratings of just 0.278. This suggests that the properties used to define categories in hand-crafted resources like WordNet are not always those that actually reflect how humans think of these categories.

6 Concluding Remarks

Much of what we understand about different categories is based on tacit and defeasible knowledge of the outside world, knowledge that cannot easily be shoe-horned into the rigid *is-a* structure of an ontology like WordNet. This already-complex picture

is complicated even further by the often metaphoric relationship between words and the categories they denote, and by the fact that the metaphor/literal distinction is not binary but gradable. Furthermore, the gradability of category membership is clearly influenced by context: in a corpus describing the exploits of Vikings, an axe will most likely be seen as a kind of weapon, but in a corpus dedicated to forestry, it will likely describe a tool. A resource like WordNet, in which *is-a* links are reserved for category relationships that are always true, in any context, is going to be inherently limited when dealing with real text.

We have described an approach that can be seen as a functional equivalent to the CPA (Corpus Pattern Analysis) approach of Pustejovsky *et al*. (2004), in which our goal is not that of automated induction of word senses in context (as it is in CPA) but the automated induction of flexible, context-sensitive category structures. As such, our goal is primarily ontological rather than lexicographic, though both approaches are complementary since each views syntagmatic evidence as the key to understanding the use of lexical concepts in context. By defining category membership in terms of syntagmatic expectations, we establish a functional and gradable basis for determining whether one lexical concept (or synset) in WordNet deserves to be seen as a descendant of another in a particular corpus and context. Augmented with ontological constraints derived from the usage of "X-like Y" patterns on the web, we also show how these membership functions can implement Glucksberg's (2001) theory of category inclusion.

We have focused on just one syntagmatic pattern here – adjectival modification of nouns – but categorization can be inferred from a wide range of productive patterns in text, particularly those concerning verbs and their case-fillers. For instance, verb-centred similes of the form "to V+*inf* like a|an N" and "to be V+*past* like a|an N" reveal insights into the diagnostic behaviour of entities (e.g., that predators hunt, that prey is hunted, that eagles soar and bombs explode). Taken together, adjective-based properties and verb-based behaviours can paint an even more comprehensive picture of each lexical concept, so that e.g., political agents that kill can be categorized as assassins, loyal entities that fight can be categorized as soldiers, and so on. An important next step, then, is to mine these behaviours from the web and incorporate the corresponding syntagmatic expectations into our category definitions. The symbolic nature of the resulting definitions means these can serve not just as mathematical membership functions, but as "active glosses", capable of recruiting their own members in a particular context while demonstrating a flexibility with categorization and a genuine competence with metaphor.

References

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating WordNet-based Measures of Lexical Semantic Relatedness. *Computational Linguistics*, 32(1), pp 13-47.

Christiane Fellbaum (ed.). 1998. *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, MA.

Cynthia Whissell. 1989. The dictionary of affect in language. In R. Plutchnik & H. Kellerman (Eds.). *Emotion: Theory and research*. New York, Harcourt Brace, 113-131.

James Pustejovsky, Patrick Hanks and Anna Rumshisky. 2004. Automated Induction of Sense in Context. In Proceedings of COLING 2004, Geneva, pp 924-931.

Patrick Hanks. 2006. Metaphoricity is a Gradable. In A. Stefanowitsch and S. Gries (eds.). *Corpora in Cognitive Linguistics. Vol. 1: Metaphor and Metonymy*. Berlin: Mouton.

Patrick Hanks. 2004. The syntagmatics of metaphor and idiom. *International Journal of Lexicography*, 17(3).

Philipp Cimiano, Andreas Hotho, and Steffen Staab. 2005. Learning Concept Hierarchies from Text Corpora using Formal Concept Analysis. *Journal of AI Research*, 24: 305-339.

Pieter De Leenheer and Aldo de Moor. 2005. Context-driven Disambiguation in Ontology Elicitation. In Shvaiko P. & Euzenat J. (eds.), Context and Ontologies: Theory, Practice and Applications, AAAI Tech Report WS-05-01. AAAI Press, pp 17-24.

Sam Glucksberg. 2001. *Understanding figurative language: From metaphors to idioms*. Oxford: Oxford University Press.

A Bayesian Model for Discovering Typological Implications

Hal Daumé III
School of Computing
University of Utah
me@hal3.name

Lyle Campbell
Department of Linguistics
University of Utah
lcampbel@hum.utah.edu

Abstract

A standard form of analysis for linguistic typology is the universal implication. These implications state facts about the range of extant languages, such as "if objects come after verbs, then adjectives come after nouns." Such implications are typically discovered by painstaking hand analysis over a small sample of languages. We propose a computational model for assisting at this process. Our model is able to discover both well-known implications as well as some novel implications that deserve further study. Moreover, through a careful application of hierarchical analysis, we are able to cope with the well-known sampling problem: languages are not independent.

1 Introduction

Linguistic typology aims to distinguish between logically possible languages and actually observed languages. A fundamental building block for such an understanding is the *universal implication* (Greenberg, 1963). These are short statements that restrict the space of languages in a concrete way (for instance "object-verb ordering implies adjective-noun ordering"); Croft (2003), Hawkins (1983) and Song (2001) provide excellent introductions to linguistic typology. We present a statistical model for automatically discovering such implications from a large typological database (Haspelmath et al., 2005).

Analyses of universal implications are typically performed by linguists, inspecting an array of 30-100 languages and a few pairs of features. Looking at all pairs of features (typically several hundred) is virtually impossible by hand. Moreover, it is insufficient to simply look at counts. For instance, results presented in the form "verb precedes object implies prepositions in 16/19 languages" are nonconclusive. While compelling, this is not enough evidence to decide if this is a statistically well-founded implication. For one, maybe 99% of languages have prepositions: then the fact that we've achieved a rate of 84% actually seems really bad. Moreover, if the 16 languages are highly related historically or areally (geographically), and the other 3 are not, then we may have only learned something about geography.

In this work, we propose a statistical model that deals cleanly with these difficulties. By building a computational model, it is possible to apply it to a very large typological database and search over many thousands of pairs of features. Our model hinges on two novel components: a statistical noise model a hierarchical inference over language families. To our knowledge, there is no prior work directly in this area. The closest work is represented by the books *Possible and Probable Languages* (Newmeyer, 2005) and *Language Classification by Numbers* (McMahon and McMahon, 2005), but the focus of these books is on automatically discovering phylogenetic trees for languages based on Indo-European cognate sets (Dyen et al., 1992).

2 Data

The database on which we perform our analysis is the *World Atlas of Language Structures* (Haspelmath et al., 2005). This database contains information about 2150 languages (sampled from across the world; Figure 1 depicts the locations of lan-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 65–72,
Prague, Czech Republic, June 2007.

Language	Numeral Classifiers	Rel/N Order	O/V Order	Glottalized Consonants	Tone	Number of Genders
English	Absent	NRel	VO	None	None	Three
Hindi	Absent	RelN	OV	None	None	Two
Mandarin	Obligatory	RelN	VO	None	Complex	None
Russian	Absent	NRel	VO	None	None	Three
Tukang Besi	Absent	?	Either	Implosives	None	Three
Zulu	Absent	NRel	VO	Ejectives	Simple	Five+

Table 1: Example database entries for a selection of diverse languages and features.

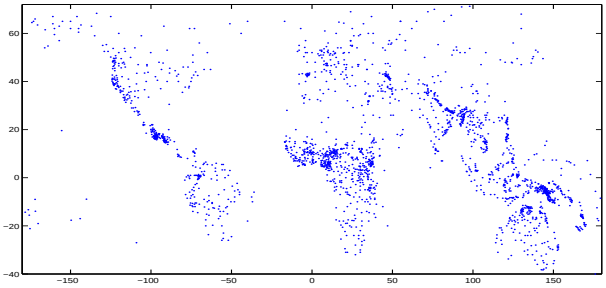


Figure 1: Map of the 2150 languages in the database.

guages). There are 139 *features* in this database, broken down into categories such as "Nominal Categories," "Simple Clauses," "Phonology," "Word Order," etc. The database is *sparse*: for many language/feature pairs, the feature value is unknown. In fact, only about 16% of all possible language/feature pairs are known. A sample of five languages and six features from the database are shown in Table 1.

Importantly, the density of samples is not random. For certain languages (eg., English, Chinese, Russian), nearly all features are known, whereas other languages (eg., Asturian, Omagua, Frisian) that have fewer than five feature values known. Furthermore, some features are known for many languages. This is due to the fact that certain features take less effort to identify than others. Identifying, for instance, if a language has a particular set of phonological features (such as glottalized consonants) requires only listening to speakers. Other features, such as determining the order of relative clauses and nouns require understanding much more of the language.

3 Models

In this section, we propose two models for automatically uncovering universal implications from noisy, sparse data. First, note that even well attested implications are not always exceptionless. A common example is that verbs preceding objects ("VO") implies adjectives following nouns ("NA"). This implication (VO $\supset$ NA) has one glaring exception: English. This is one particular form of noise. Another source of noise stems from transcription. WALS contains data about languages documented by field linguists as early as the 1900s. Much of this older data was collected before there was significant agreement in documentation style. Different field linguists often had different dimensions along which they segmented language features into classes. This leads to noise in the properties of individual languages.

Another difficulty stems from the *sampling problem*. This is a well-documented issue (see, eg., (Croft, 2003)) stemming from the fact that any set of languages is not sampled uniformly from the space of all probable languages. Politically interesting languages (eg., Indo-European) and typologically unusual languages (eg., Dyirbal) are better documented than others. Moreover, languages are not independent: German and Dutch are more similar than German and Hindi due to history and geography.

The first model, FLAT, treats each language as independent. It is thus susceptible to sampling problems. For instance, the WALS database contains a half dozen versions of German. The FLAT model considers these versions of German just as statistically independent as, say, German and Hindi. To cope with this problem, we then augment the FLAT model into a HIERarchical model that takes advantage of known hierarchies in linguistic phylogenetics. The HIER model explicitly models the fact that individual languages are *not* independent and exhibit strong familial dependencies. In both models, we initially restrict our attention to pairs of features. We will describe our models as if all features are binary. We expand any multi-valued feature with K values into K binary features in a "one versus rest" manner.

3.1 The FLAT Model

In the FLAT model, we consider a $2 \times N$ matrix of feature values. The N corresponds to the number of languages, while the 2 corresponds to the two features currently under consideration (eg., object/verb order and noun/adjective order). The order of the

two features is important: f_1 implies f_2 is logically different from f_2 implies f_1. Some of the entries in the matrix will be unknown. We may safely remove all languages from consideration for which *both* are unknown, but we do *not* remove languages for which only one is unknown. We do so because our model needs to capture the fact that if f_2 is *always* true, then $f_1 \supset f_2$ is uninteresting.

The statistical model is set up as follows. There is a single variable (we will denote this variable "m") corresponding to whether the implication holds. Thus, $m = 1$ means that f_1 implies f_2 and $m = 0$ means that it does not. Independent of m, we specify two feature priors, π_1 and π_2 for f_1 and f_2 respectively. π_1 specifies the prior probability that f_1 will be true, and π_2 specifies the prior probability that f_2 will be true. One can then put the model together naïvely as follows. If $m = 0$ (i.e., the implication does not hold), then the entire data matrix is generated by choosing values for f_1 (resp., f_2) independently according to the prior probability π_1 (resp., π_2). On the other hand, if $m = 1$ (i.e., the implication *does* hold), then the first column of the data matrix is generated by choosing values for f_1 independently by π_1, but the second column is generated differently. In particular, if for a particular language, we have that f_1 is true, then the fact that the implication holds means that f_2 *must* be true. On the other hand, if f_1 is false for a particular language, then we may generate f_2 according to the prior probability π_2. Thus, having $m = 1$ means that the model is significantly more constrained. In equations:

$$p(f_1 \mid \pi_1) = \pi_1^{f_1}(1-\pi_1)^{1-f_1}$$

$$p(f_2 \mid f_1, \pi_2, m) = \begin{cases} f_2 & m = f_1 = 1 \\ \pi_2^{f_2}(1-\pi_2)^{1-f_2} & \text{otherwise} \end{cases}$$

The problem with this naïve model is that it does not take into account the fact that there is "noise" in the data. (By noise, we refer either to mis-annotations, or to "strange" languages like English.) To account for this, we introduce a simple noise model. There are several options for parameterizing the noise, depending on what independence assumptions we wish to make. One could simply specify a noise rate for the entire data set. One could alternatively specify a language-specific noise rate. Or one could specify a feature-specific noise rate. We opt for a blend between the first and second option. We assume an underlying noise rate for the entire data set, but that, conditioned on this underlying rate, there is a language-specific noise level. We believe this to be an appropriate noise model because it models the fact that the majority of information for a single language is from a single source. Thus, if there is an error in the database, it is more likely that other errors will be for the same languages.

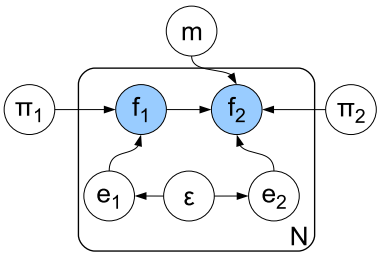


Figure 2: Graphical model for the FLAT model.

In order to model this statistically, we assume that there are latent variables $e_{1,n}$ and $e_{2,n}$ for each language n. If $e_{1,n} = 1$, then the first feature for language n is wrong. Similarly, if $e_{2,n} = 1$, then the second feature for language n is wrong. Given this model, the probabilities are exactly as in the naïve model, with the exception that instead of using f_1 (resp., f_2), we use the exclusive-or[1] $f_1 \otimes e_1$ (resp., $f_2 \otimes e_2$) so that the feature values are flipped whenever the noise model suggests an error.

The graphical model for the FLAT model is shown in Figure 2. Circular nodes denote random variables and arrows denote conditional dependencies. The rectangular plate denotes the fact that the elements contained within it are replicated N times (N is the number of languages). In this model, there are four "root" nodes: the implication value m; the two feature prior probabilities π_1 and π_2; and the language-specific error rate ϵ. On all of these nodes we place Bayesian priors. Since m is a binary random variable, we place a Bernoulli prior on it. The πs are Bernoulli random variables, so they are given independent Beta priors. Finally, the noise rate ϵ is also given a Beta prior. For the two Beta parameters governing the error rate (i.e., a_ϵ and b_ϵ) we set these by hand so that the mean expected error rate is 5% and the probability of the error rate being between 0% and 10% is 50% (this number is based on an expert opinion of the noise-rate in the data). For the rest of

[1]The exclusive-or of a and b, written $a \otimes b$, is true exactly when either a or b is true but not both.

the parameters we use uniform priors.

3.2 The HIER Model

A significant difficulty in working with any large typological database is that the languages will be sampled *non*uniformly. In our case, this means that implications that seem true in the FLAT model may only be true for, say, Indo-European, and the remaining languages are considered noise. While this may be interesting in its own right, we are more interested in discovering implications that are truly universal.

We model this using a hierarchical Bayesian model. In essence, we take the FLAT model and build a notion of language relatedness into it. In particular, we enforce a hierarchy on the m implication variables. For simplicity, suppose that our "hierarchy" of languages is nearly flat. Of the N languages, half of them are Indo-European and the other half are Austronesian. We will use a nearly identical model to the FLAT model, but instead of having a single m variable, we have three: one for IE, one for Austronesian and one for "all languages."

For a general tree, we assign one implication variable for each node (including the root and leaves). The goal of the inference is to infer the value of the m variable corresponding to the root of the tree.

All that is left to specify the full HIER model is to specify the probability distribution of the m random variables. We do this as follows. We place a zero mean Gaussian prior with (unknown) variance σ^2 on the root m. Then, for a non-root node, we use a Gaussian with mean equal to the "m" value of the parent and tied variance σ^2. In our three-node example, this means that the root is distributed $\mathcal{N}or(0, \sigma^2)$ and each child is distributed $\mathcal{N}or(m_{\text{root}}, \sigma^2)$, where m_{root} is the random variable corresponding to the root. Finally, the leaves (corresponding to the languages themselves) are distributed *logistic-binomial*. Thus, the m random variable corresponding to a leaf (language) is distributed $\mathcal{B}in(s(m_{\text{par}}))$, where m_{par} is the m value for the parent (internal) node and s is the sigmoid function $s(x) = [1 + exp(-x)]^{-1}$.

The intuition behind this model is that the m value at each node in the tree (where a node is either "all languages" or a specific language family or an individual language) specifies the extent to which the implication under consideration holds for that node. A large positive m means that the implication is very likely to hold. A large negative value means it is very likely to not hold. The normal distributions across edges in the tree indicate that we expect the m values not to change too much across the tree. At the leaves (i.e., individual languages), the logistic-binomial simply transforms the real-valued ms into the range $[0, 1]$ so as to make an appropriate input to the binomial distribution.

4 Statistical Inference

In this section, we describe how we use Markov chain Monte Carlo methods to perform inference in the statistical models described in the previous section; Andrieu et al. (2003) provide an excellent introduction to MCMC techniques. The key idea behind MCMC techniques is to approximate intractable expectations by drawing random samples from the probability distribution of interest. The expectation can then be approximated by an empirical expectation over these sample.

For the FLAT model, we use a combination of Gibbs sampling with rejection sampling as a subroutine. Essentially, all sampling steps are standard Gibbs steps, except for sampling the error rates e. The Gibbs step is not available analytically for these. Hence, we use rejection sampling (drawing from the Beta prior and accepting according to the posterior).

The sampling procedure for the HIER model is only slightly more complicated. Instead of performing a simple Gibbs sample for m in Step (4), we first sample the m values for the internal nodes using simple Gibbs updates. For the leaf nodes, we use rejection sampling. For this rejection, we draw proposal values from the Gaussian specified by the parent m, and compute acceptance probabilities.

In all cases, we run the outer Gibbs sampler for 1000 iterations and each rejection sampler for 20 iterations. We compute the marginal values for the m implication variables by averaging the sampled values after dropping 200 "burn-in" iterations.

5 Data Preprocessing and Search

After extracting the raw data from the WALS electronic database (Haspelmath et al., 2005)[2], we perform a minor amount of preprocessing. Essentially, we have manually removed certain feature

[2]This is nontrivial—we are currently exploring the possibility of freely sharing these data.

values from the database because they are underrepresented. For instance, the "Glottalized Consonants" feature has eight possible values (one for "none" and seven for different varieties of glottalized consonants). We reduce this to simply two values "has" or "has not." 313 languages have no glottalized consonants and 139 have some variety of glottalized consonant. We have done something similar with approximately twenty of the features.

For the HIER model, we obtain the hierarchy in one of two ways. The first hierarchy we use is the "linguistic hierarchy" specified as part of the WALS data. This hierarchy divides languages into families and subfamilies. This leads to a tree with the leaves at depth four. The root has 38 immediate children (corresponding to the major families), and there are a total of 314 internal nodes. The second hierarchy we use is an areal hierarchy obtained by clustering languages according to their latitude and longitude. For the clustering we first cluster all the languages into 6 "macro-clusters." We then cluster each macro-cluster individually into 25 "micro-clusters." These micro-clusters then have the languages at their leaves. This yields a tree with 31 internal nodes.

Given the database (which contains approximately 140 features), performing a raw search even over all possible *pairs* of features would lead to over $19,000$ computations. In order to reduce this space to a more manageable number, we filter:

- There must be at least 250 languages for which *both* features are known.
- There must be at least 15 languages for which both feature values hold simultaneously.
- Whenever f_1 is true, at least half of the languages also have f_2 true.

Performing all these filtration steps reduces the number of pairs under consideration to 3442. While this remains a computationally expensive procedure, we were able to perform all the implication computations for these 3442 possible pairs in about a week on a single modern machine (in Matlab).

6 Results

The task of discovering universal implications is, at its heart, a data-mining task. As such, it is difficult to evaluate, since we often do not know the correct answers! If our model only found well-documented implications, this would be interesting but useless from the perspective of aiding linguists focus their energies on new, plausible implications. In this section, we present the results of our method, together with both a quantitative and qualitative evaluation.

6.1 Quantitative Evaluation

In this section, we perform a quantitative evaluation of the results based on *predictive power*. That is, one generally would prefer a system that finds implications that hold with high probability across the data. The word "generally" is important: this quality is neither necessary nor sufficient for the model to be good. For instance, finding 1000 implications of the form $A_1 \supset X, A_2 \supset X, \ldots, A_{1000} \supset X$ is completely uninteresting if X is true in 99% of the cases. Similarly, suppose that a model can find 1000 implications of the form $X \supset A_1, \ldots, X \supset A_{1000}$, but X is only true in five languages. In both of these cases, according to a "predictive power" measure, these would be ideal systems. But they are both somewhat uninteresting.

Despite these difficulties with a predictive power-based evaluation, we feel that it is a good way to understand the relative merits of our different models. Thus, we compare the following systems: FLAT (our proposed flat model), LINGHIER (our model using the phylogenetic hierarchy), DISTHIER (our model using the areal hierarchy) and RANDOM (a model that ranks implications—that meet the three qualifications from the previous section—randomly).

The models are scored as follows. We take the entire WALS data set and "hide" a random 10% of the entries. We then perform full inference and ask the inferred model to predict the missing values. The accuracy of the model is the accuracy of its predictions. To obtain a sense of the quality of the ranking, we perform this computation on the top k ranked implications provided by each model; $k \in \{2, 4, 8, \ldots, 512, 1024\}$.

The results of this quantitative evaluation are shown in Figure 3 (on a log-scale for the x-axis). The two best-performing models are the two hierarchical models. The flat model does significantly worse and the random model does terribly. The vertical lines are a standard deviation over 100 folds of the experiment (hiding a different 10% each time). The difference between the two hierarchical models is typically not statistically significant. At the top of the ranking, the model based on phylogenetic

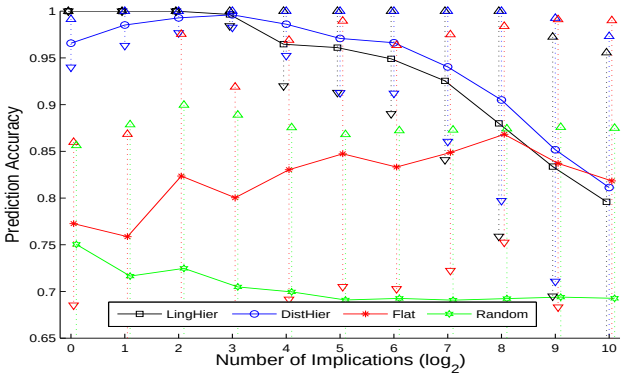


Figure 3: Results of quantitative (predictive) evaluation. Top curves are the hierarchical models; middle is the flat model; bottom is the random baseline.

information performs marginally better; at the bottom of the ranking, the order flips. Comparing the hierarchical models to the flat model, we see that adequately modeling the *a priori* similarity between languages is quite important.

6.2 Cross-model Comparison

The results in the previous section support the conclusion that the two hierarchical models are doing something significantly different (and better) than the flat model. This clearly must be the case. The results, however, do not say whether the two hierarchies are substantially different. Moreover, are the results that they produce substantially different. The answer to these two questions is "yes."

We first address the issue of tree similarity. We consider all pairs of languages which are at distance 0 in the areal tree (i.e., have the same parent). We then look at the mean tree-distance between those languages in the phylogenetic tree. We do this for all distances in the areal tree (because of its construction, there are only three: 0, 2 and 4). The mean distances in the phylogenetic tree corresponding to these three distances in the areal tree are: 2.9, 3.5 and 4.0, respectively. This means that languages that are "nearby" in the areal tree are quite often very far apart in the phylogenetic tree.

To answer the issue of whether the results obtained by the two trees are similar, we employ Kendall's τ statistic. Given two ordered lists, the τ statistic computes how correlated they are. τ is always between 0 and 1, with 1 indicating identical ordering and 0 indicated completely reversed ordering. The results are as follows. Comparing FLAT to LINGHIER yield $\tau = 0.4144$, a very low correlation. Between FLAT and DISTHIER, $\tau = 0.5213$, also very low. These two are as expected. Finally, between LINGHIER and DISTHIER, we obtain $\tau = 0.5369$, a very low correlation, considering that both perform well predictively.

6.3 Qualitative Analysis

For the purpose of a qualitative analysis, we reproduce the top 30 implications discovered by the LINGHIER model in Table 2 (see the final page).[3] Each implication is numbered, then the actual implication is presented. For instance, #7 says that any language that has adjectives preceding their governing nouns also has numerals preceding their nouns. We additionally provide an "analysis" of many of these discovered implications. Many of them (eg., #7) are well known in the typological literature. These are simply numbered according to well-known references. For instance our #7 is implication #18 from Greenberg, reproduced by Song (2001). Those that reference Hawkins (eg., #11) are based on implications described by Hawkins (1983); those that reference Lehmann are references to the principles decided by Lehmann (1981) in Ch 4 & 8.

Some of the implications our model discovers are obtained by composition of well-known implications. For instance, our #3 (namely, OV $\supset$ Genitive-Noun) can be obtained by combining Greenberg #4 (OV $\supset$ Postpositions) and Greenberg #2a (Postpositions $\supset$ Genitive-Noun). It is quite encouraging that 14 of our top 21 discovered implications are well-known in the literature (and this, not even considering the tautalogically true implications)! This strongly suggests that our model is doing something reasonable and that there is true structure in the data.

In addition to many of the known implications found by our model, there are many that are "unknown." Space precludes attempting explanations of them all, so we focus on a few. Some are easy. Consider #8 (Strongly suffixing $\supset$ Tense-aspect suffixes): this is quite plausible—if you have a lan-

[3] In truth, our model discovers several tautalogical implications that we have removed by hand before presentation. These are examples like "SVO $\supset$ VO" or "No unusual consonants $\supset$ no glottalized consonants." It is, of course, good that our model discovers these, since they are obviously true. However, to save space, we have withheld them from presentation here. The 30th implication presented here is actually the 83rd in our full list.

guage that tends to have suffixes, it will probably have suffixes for tense/aspect. Similarly, #10 states that languages with verb morphology for questions lack question particles; again, this can be easily explained by an appeal to economy.

Some of the discovered implications require a more involved explanation. One such example is #20: labial-velars implies no uvulars.[4] It turns out that labial-velars are most common in Africa just north of the equator, which is also a place that has very few uvulars (there are a handful of other examples, mostly in Papua New Guinea). While this implication has not been previously investigated, it makes some sense: if a language has one form of rare consonant, it is unlikely to have another.

As another example, consider #28: Obligatory suffix pronouns implies no possessive affixes. This means is that in languages (like English) for which pro-drop is impossible, possession is not marked morphologically on the head noun (like English, "book" appears the same regarless of if it is "his book" or "the book"). This also makes sense: if you cannot drop pronouns, then one usually will mark possession on the pronoun, not the head noun. Thus, you do not need marking on the head noun.

Finally, consider #25: High and mid front vowels (i.e., / u/, etc.) implies large vowel inventory ($\geq$ 7 vowels). This is supported by typological evidence that high and mid front vowels are the "last" vowels to be added to a language's repertoire. Thus, in order to get them, you must also have many other types of vowels already, leading to a large vowel inventory.

Not all examples admit a simple explanation and are worthy of further thought. Some of which (like the ones predicated on "SV") may just be peculiarities of the annotation style: the subject verb order changes frequently between transitive and intransitive usages in many languages, and the annotation reflects just one. Some others are bizzarre: why not having fricatives should mean that you don't have tones (#27) is not a priori clear.

6.4 Multi-conditional Implications

Many implications in the literature have multiple implicants. For instance, much research has gone into looking at which implications hold, considering only "VO" languages, or considering only languages with prepositions. It is straightforward to modify our model so that it searches over triples of features, conditioning on two and predicting the third. Space precludes an in-depth discussion of these results, but we present the top examples in Table 3 (after removing the tautalogically true examples, which are more numerous in this case, as well as examples that are directly obtainable from Table 2). It is encouraging that in the top 1000 multi-conditional implications found, the most frequently used were "OV" (176 times) "Postpositions" (157 times) and "Adjective-Noun" (89 times). This result agrees with intuition.

Implicants		Implicand
Postpositions Adjective-Noun	$\supset$	Demonstrative-Noun
Posessive prefixes Tense-aspect suffixes	$\supset$	Genitive-Noun
Case suffixes Plural suffix	$\supset$	Genitive-Noun
Adjective-Noun Genitive-Noun	$\supset$	OV
High cons/vowel ratio No front-rounded vowels	$\supset$	No tones
Negative affix Genitive-Noun	$\supset$	OV
No front-rounded vowels Labial velars	$\supset$	Large vowel quality inventory
Subordinating suffix Tense-aspect suffixes	$\supset$	Postpositions
No case affixes Prepositions	$\supset$	Initial subordinator word
Strongly suffixing Plural suffix	$\supset$	Genitive-Noun

Table 3: Top implications discovered by the LINGHIER multi-conditional model.

7 Discussion

We have presented a Bayesian model for discovering universal linguistic implications from a typological database. Our model is able to account for noise in a linguistically plausible manner. Our hierarchical models deal with the sampling issue in a unique way, by using prior knowledge about language families to "group" related languages. Quantitatively, the hierarchical information turns out to be quite useful, regardless of whether it is phylogenetically- or areally-based. Qualitatively, our model can recover many well-known implications as well as many more potential implications that can be the object of future linguistic study. We believe that our model is suf-

[4] Labial-velars and uvulars are rare consonants (order 100 languages). Labial-velars are joined sounds like /kp/ and /gb/ (to English speakers, sounding like chicken noises); uvulars sounds are made in the back of the throat, like snoring.

#	Implicant ⊃ Implicand	Analysis
1	Postpositions ⊃ Genitive-Noun	Greenberg #2a
2	OV ⊃ Postpositions	Greenberg #4
3	OV ⊃ Genitive-Noun	Greenberg #4 + Greenberg #2a
4	Genitive-Noun ⊃ Postpositions	Greenberg #2a (converse)
5	Postpositions ⊃ OV	Greenberg #2b (converse)
6	SV ⊃ Genitive-Noun	???
7	Adjective-Noun ⊃ Numeral-Noun	Greenberg #18
8	Strongly suffixing ⊃ Tense-aspect suffixes	Clear explanation
9	VO ⊃ Noun-Relative Clause	Lehmann
10	Interrogative verb morph ⊃ No question particle	Appeal to economy
11	Numeral-Noun ⊃ Demonstrative-Noun	Hawkins XVI (for postpositional languages)
12	Prepositions ⊃ VO	Greenberg #3 (converse)
13	Adjective-Noun ⊃ Demonstrative-Noun	Greenberg #18
14	Noun-Adjective ⊃ Postpositions	Lehmann
15	SV ⊃ Postpositions	???
16	VO ⊃ Prepositions	Greenberg #3
17	Initial subordinator word ⊃ Prepositions	Operator-operand principle (Lehmann)
18	Strong prefixing ⊃ Prepositions	Greenberg #27b
19	Little affixation ⊃ Noun-Adjective	???
20	Labial-velars ⊃ No uvular consonants	See text
21	Negative word ⊃ No pronominal possessive affixes	See text
22	Strong prefixing ⊃ VO	Lehmann
23	Subordinating suffix ⊃ Strongly suffixing	???
24	Final subordinator word ⊃ Postpositions	Operator-operand principle (Lehmann)
25	High and mid front vowels ⊃ Large vowel inventories	See text
26	Plural prefix ⊃ Noun-Genitive	???
27	No fricatives ⊃ No tones	???
28	Obligatory subject pronouns ⊃ No pronominal possessive affixes	See text
29	Demonstrative-Noun ⊃ Tense-aspect suffixes	Operator-operand principle (Lehmann)
30	Prepositions ⊃ Noun-Relative clause	Lehmann, Hawkins

Table 2: Top 30 implications discovered by the LINGHIER model.

ficiently general that it could be applied to many different typological databases — we attempted not to "overfit" it to WALS. Our hope is that the automatic discovery of such implications not only aid typologically-inclined linguists, but also other groups. For instance, well-attested universal implications have the potential to reduce the amount of data field linguists need to collect. They have also been used computationally to aid in the learning of unsupervised part of speech taggers (Schone and Jurafsky, 2001). Many extensions are possible to this model; for instance attempting to uncover typologically hierarchies and other higher-order structures. We have made the full output of all models available at `http://hal3.name/WALS`.

Acknowledgments. We are grateful to Yee Whye Teh, Eric Xing and three anonymous reviewers for their feedback on this work.

References

Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. 2003. An introduction to MCMC for machine learning. *Machine Learning (ML)*, 50:5–43.

William Croft. 2003. *Typology and Univerals*. Cambridge University Press.

Isidore Dyen, Joseph Kurskal, and Paul Black. 1992. An Indoeuropean classification: A lexicostatistical experiment. *Transactions of the American Philosophical Society*, 82(5). American Philosophical Society.

Joseph Greenberg, editor. 1963. *Universals of Languages*. MIT Press.

Martin Haspelmath, Matthew Dryer, David Gil, and Bernard Comrie, editors. 2005. *The World Atlas of Language Structures*. Oxford University Press.

John A. Hawkins. 1983. *Word Order Universals: Quantitative analyses of linguistic structure*. Academic Press.

Winfred Lehmann, editor. 1981. *Syntactic Typology*, volume xiv. University of Texas Press.

April McMahon and Robert McMahon. 2005. *Language Classification by Numbers*. Oxford University Press.

Frederick J. Newmeyer. 2005. *Possible and Probable Languages: A Generative Perspective on Linguistic Typology*. Oxford University Press.

Patrick Schone and Dan Jurafsky. 2001 *Language Independent Induction of Part of Speech Class Labels Using only Language Universals*. Machine Learning: Beyond Supervision.

Jae Jung Song. 2001. *Linguistic Typology: Morphology and Syntax*. Longman Linguistics Library.

A Discriminative Language Model with Pseudo-Negative Samples

Daisuke Okanohara† **Jun'ichi Tsujii**†‡§
†Department of Computer Science, University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo, Japan
‡School of Informatics, University of Manchester
§NaCTeM (National Center for Text Mining)
{hillbig,tsujii}@is.s.u-tokyo.ac.jp

Abstract

In this paper, we propose a novel discriminative language model, which can be applied quite generally. Compared to the well known N-gram language models, discriminative language models can achieve more accurate discrimination because they can employ overlapping features and non-local information. However, discriminative language models have been used only for re-ranking in specific applications because negative examples are not available. We propose sampling pseudo-negative examples taken from probabilistic language models. However, this approach requires prohibitive computational cost if we are dealing with quite a few features and training samples. We tackle the problem by estimating the latent information in sentences using a semi-Markov class model, and then extracting features from them. We also use an online margin-based algorithm with efficient kernel computation. Experimental results show that pseudo-negative examples can be treated as real negative examples and our model can classify these sentences correctly.

1 Introduction

Language models (LMs) are fundamental tools for many applications, such as speech recognition, machine translation and spelling correction. The goal of LMs is to determine whether a sentence is correct or incorrect in terms of grammars and pragmatics.

The most widely used LM is a probabilistic language model (PLM), which assigns a probability to a sentence or a word sequence. In particular, N-grams with maximum likelihood estimation (NLMs) are often used. Although NLMs are simple, they are effective for many applications.

However, NLMs cannot determine correctness of a sentence independently because the probability depends on the length of the sentence and the global frequencies of each word in it. For example, $p(S_1) < p(S_2)$, where $p(S)$ is the probability of a sentence S given by an NLM, does not always mean that S_2 is more correct, but instead could occur when S_2 is shorter than S_1, or if S_2 has more common words than S_1. Another problem is that NLMs cannot handle overlapping information or non-local information easily, which is important for more accurate sentence classification. For example, a NLM could assign a high probability to a sentence even if it does not have a verb.

Discriminative language models (DLMs) have been proposed to classify sentences directly as correct or incorrect (Gao et al., 2005; Roark et al., 2007), and these models can handle both non-local and overlapping information. However DLMs in previous studies have been restricted to specific applications. Therefore the model cannot be used for other applications. If we had negative examples available, the models could be trained directly by discriminating between correct and incorrect sentences.

In this paper, we propose a generic DLM, which can be used not only for specific applications, but also more generally, similar to PLMs. To achieve

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 73–80,
Prague, Czech Republic, June 2007.

this goal, we need to solve two problems. The first is that since we cannot obtain negative examples (incorrect sentences), we need to generate them. The second is the prohibitive computational cost because the number of features and examples is very large. In previous studies this problem did not arise because the amount of training data was limited and they did not use a combination of features, and thus the computational cost was negligible.

To solve the first problem, we propose sampling incorrect sentences taken from a PLM and then training a model to discriminate between correct and incorrect sentences. We call these examples Pseudo-Negative because they are not actually negative sentences. We call this method DLM-PN (DLM with Pseudo-Negative samples).

To deal with the second problem, we employ an online margin-based learning algorithm with fast kernel computation. This enables us to employ combinations of features, which are important for discrimination between correct and incorrect sentences. We also estimate the latent information in sentences by using a semi-Markov class model to extract features. Although there are substantially fewer latent features than explicit features such as words or phrases, latent features contain essential information for sentence classification.

Experimental results show that these pseudo-negative samples can be treated as incorrect examples, and that DLM-PN can learn to correctly discriminate between correct and incorrect sentences and can therefore classify these sentences correctly.

2 Previous work

Probabilistic language models (PLMs) estimate the probability of word strings or sentences. Among these models, N-gram language models (NLMs) are widely used. NLMs approximate the probability by conditioning only on the preceding $N-1$ words. For example, let S denote a sentence of t words, $S := w_1, w_2, \ldots, w_t$. Then, by the chain rule of probability and the approximation, we have

$$\begin{aligned} P(S) &= P(w_1, w_2, \ldots, w_t) \\ &= \prod_{i=1\ldots t} P(w_i | w_{i-N+1}, \ldots, w_{i-1}). \quad (1) \end{aligned}$$

The parameters can be estimated using the maximum likelihood method.

Since the number of parameters in NLM is still large, several smoothing methods are used (Chen and Goodman, 1998) to produce more accurate probabilities, and to assign nonzero probabilities to any word string.

However, since the probabilities in NLMs depend on the length of the sentence, two sentences of different length cannot be compared directly.

Recently, Whole Sentence Maximum Entropy Models (Rosenfeld et al., 2001) (WSMEs) have been introduced. They assign a probability to each sentence using a maximum entropy model. Although WSMEs can encode all features of a sentence including non-local ones, they are only slightly superior to NLMs, in that they have the disadvantage of being computationally expensive, and not all relevant features can be included.

A discriminative language model (DLM) assigns a score $f(S)$ to a sentence S, measuring the correctness of a sentence in terms of grammar and pragmatics, so that $f(S) > 0$ implies S is correct and $f(S) < 0$ implies S is incorrect. A PLM can be considered as a special case of a DLM by defining f using $P(S)$. For example, we can take $f(S) = P(S)/|S| - \alpha$, where α is some threshold, and $|S|$ is the length of S.

Given a sentence S, we extract a feature vector ($\phi(S)$) from it using a pre-defined set of feature functions $\{\phi_j\}_{j=1}^m$. The form of the function f we use is

$$f(S) = \mathbf{w} \cdot \phi(S), \quad (2)$$

where $\mathbf{w}$ is a feature weighting vector.

Since there is no restriction in designing $\phi(S)$, DLMs can make use of both over-lapping and non-local information in S. We estimate $\mathbf{w}$ using training samples $\{(S_i, y_i)\}$ for $i = 1...t$, where $y_i = 1$ if S_i is correct and $y_i = -1$ if S_i is incorrect.

However, it is hard to obtain incorrect sentences because only correct sentences are available from the corpus. This problem was not an issue for previous studies because they were concerned with specific applications and therefore were able to obtain real negative examples easily. For example, Roark (2007) proposed a discriminative language model, in which a model is trained so that a correct sentence should have higher score than others. The difference between their approach and ours is that we do not assume just one application. Moreover, they had

```
For i=1,2,...
   Choose a word w_i at random
    according to the distribution
    P(w_i|w_{i-N+1},...,w_{i-1})
   If w_i = "end of a sentence"
      Break
   End End
```

Figure 1: Sample procedure for pseudo-negative examples taken from N-gram language models.

training sets consisting of one correct sentence and many incorrect sentences, which were very similar because they were generated by the same input. Our framework does not assume any such training sets, and we treat correct or incorrect examples independently in training.

3 Discriminative Language Model with Pseudo-Negative samples

We propose a novel discriminative language model; a Discriminative Language Model with Pseudo-Negative samples (DLM-PN). In this model, pseudo-negative examples, which are all assumed to be incorrect, are sampled from PLMs.

First a PLM is built using training data and then examples, which are almost all negative, are sampled independently from PLMs. DLMs are trained using correct sentences from a corpus and negative examples from a Pseudo-Negative generator.

An advantage of sampling is that as many negative examples can be collected as correct ones, and a distinction can be clearly made between truly correct sentences and incorrect sentences, even though the latter might be correct in a local sense.

For sampling, any PLMs can be used as long as the model supports a sentence sampling procedure. In this research we used NLMs with interpolated smoothing because such models support efficient sentence sampling. Figure 1 describes the sampling procedure and figure 2 shows an example of a pseudo-negative sentence.

Since the focus is on discriminating between correct sentences from a corpus and incorrect sentences sampled from the NLM, DLM-PN may not able to classify incorrect sentences that are not generated from the NLM. However, this does not result in a serious problem, because these sentences, if they exist, can be filtered out by NLMs.

```
We know of no program, and animated
discussions about prospects for trade
barriers or regulations on the rules
of the game as a whole, and elements
of decoration of this peanut-shaped
to priorities tasks across both target
countries
```

Figure 2: Example of a sentence sampled by PLMs (Trigram).

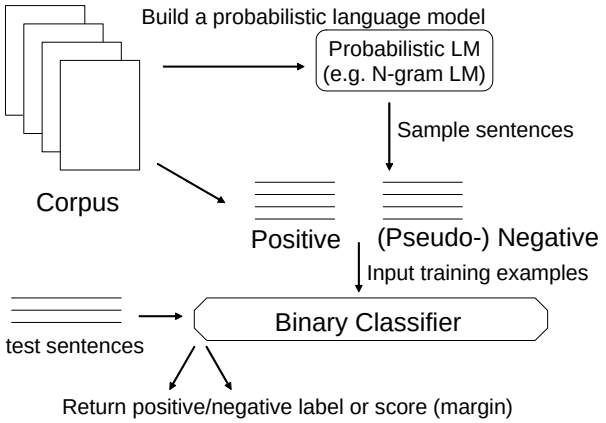


Figure 3: Framework of our classification process.

4 Online margin-based learning with fast kernel computation

The DLM-PN can be trained by using any binary classification learning methods. However, since the number of training examples is very large, batch training has suffered from prohibitively large computational cost in terms of time and memory. Therefore we make use of an online learning algorithm proposed by (Crammer et al., 2006), which has a much smaller computational cost. We follow the definition in (Crammer et al., 2006).

The initiation vector $\mathbf{w}_1$ is initialized to $\mathbf{0}$ and for each round the algorithm observes a training example $\mathbf{x}_i := \phi(S_i)$ and predicts its label y_i' to be either $+1$ or -1. After the prediction is made, the true label y_i is revealed and the algorithm suffers an *instantaneous hinge-loss* $l(\mathbf{w}; (\mathbf{x}_i, y_i)) = 1 - y_i(\mathbf{w}_i \cdot \mathbf{x}_i)$ which reflects the degree to which its prediction was wrong. If the prediction was wrong, the parameter

$\mathbf{w}$ is updated as

$$\mathbf{w}_{i+1} = argmin_{\mathbf{w}} \frac{1}{2}||\mathbf{w} - \mathbf{w}_i||^2 + C\xi \quad (3)$$

$$\text{subject to } l(\mathbf{w}; (\mathbf{x}_i, y_i)) \leq \xi \text{ and } \xi \geq 0, \quad (4)$$

where ξ is a slack term and C is a positive parameter which controls the influence of the slack term on the objective function. A large value of C will result in a more aggressive update step. This has a closed form solution as

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \tau_i y_i \mathbf{x}_i \quad (5)$$

where $\tau_i = \min\{C, \frac{l_i}{||\mathbf{x}_i||^2}\}$. As in SVMs, a final weight vector can be represented as a kernel-dependent combination of the stored training examples.

$$\mathbf{w} \cdot \mathbf{x} = \sum_i \tau_i y_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle \quad (6)$$

Using this formulation the inner product can be replaced with a general Mercer kernel $K(\mathbf{x}_i, \mathbf{x})$ such as a polynomial kernel or a Gaussian kernel.

The combination of features, which can capture correlation information, is important in DLMs. If the kernel-trick (Taylor and Cristianini, 2004) is applied to online margin-based learning, a subset of the observed examples, called the active set, needs to be stored. However in contrast to the support set in SVMs, an example is added to the active set every time the online algorithm makes a prediction mistake or when its confidence in a prediction is inadequately low. Therefore the active set can increase in size significantly and thus the total computational cost becomes proportional to the square of the number of training examples. Since the number of training examples is very large, the computational cost is prohibitive even if we apply the kernel trick.

The calculation of the inner product between two examples can be done by intersection of the activated features in each example. This is similar to a merge sort and can be executed in $O(M)$ time where M is the average number of activated features in an example. When the number of examples in the active set is A, the total computational cost is $O(M \cdot A)$. For fast kernel computation, the Polynomial Kernel Inverted method (PKI)) is proposed (Kudo and Matsumoto, 2003), which is an extension of Inverted Index in Information Retrieval. This algorithm uses a table $h(f_i)$ for each feature item, which stores examples where a feature f_i is fired. Let B be the average of $|h(f_i)|$ over all feature item. Then the kernel computation can be performed in $O(M \cdot B)$ time which is much less than the normal kernel computation time when $B \ll A$. We can easily extend this algorithm into the online setting by updating $h(f_i)$ when an observed example is added to an active set.

5 Latent features by semi-Markov class model

Another problem for DLMs is that the number of features becomes very large, because all possible N-grams are used as features. In particular, the memory requirement becomes a serious problem because quite a few active sets with many features have to be stored, not only at training time, but also at classification time. One way to deal with this is to filter out low-confidence features, but it is difficult to decide which features are important in online learning. For this reason we cluster similar N-grams using a semi-Markov class model.

The class model was originally proposed by (Martin et al., 1998). In the class model, deterministic word-to-class mappings are estimated, keeping the number of classes much smaller than the number of distinct words. A semi-Markov class model (SMCM) is an extended version of the class model, a part of which was proposed by (Deligne and BIMBOT, 1995). In SMCM, a word sequence is partitioned into a variable-length sequence of chunks and then chunks are clustered into classes (Figure 4). How a chunk is clustered depends on which chunks are adjacent to it.

The probability of a sentence $P(w_1, \ldots, w_t)$, in a bi-gram class model is calculated by

$$\prod_i P(w_{i+1}|c_{i+1}) P(c_{i+1}|c_i). \quad (7)$$

On the other hand, the probabilities in a bi-gram semi-Markov class model are calculated by

$$\sum_s \prod_i P(c_i|c_{i-1}) \cdot P(w_{t(i),t(i)+1,\ldots,u(i)}|c_i). \quad (8)$$

where s varies over all possible partitions of S, $t(i)$ and $u(i)$ denote the start and end positions respectively of the i-th chunk in partition s, and $t(i+1) =$

$u(i) + 1$ for all i. Note that each word or variable-length chunk belongs to only one class, in contrast to a hidden Markov model where each word can belong to several classes.

Using a training corpus, the mapping is estimated by maximum likelihood estimation. The log likelihood of the training corpus $(w_1, \ldots, w_n)$ in a bigram class model can be calculated as

$$\log \prod_i P(w_{i+1}|w_i) \quad (9)$$

$$= \sum_i \log P(w_{i+1}|c_{i+1})P(c_{i+1}|c_i) \quad (10)$$

$$= \sum_{c_1,c_2} F(c_1,c_2) \log \frac{F(c_1,c_2)}{F(c_1)F(c_2)} + \sum_w F(w) \log F(w). \quad (11)$$

where $F(w)$, $F(c)$ and $F(c_1, c_2)$ are frequencies of a word w, a class c and a class bi-gram c_1, c_2 in the training corpus. In (11) only the first term is used, since the second term does not depend on the class allocation. The class allocation problem is solved by an exchange algorithm as follows. First, all words are assigned to a randomly determined class. Next, for each word w, we move it to the class c for which the log-likelihood is maximized. This procedure is continued until the log-likelihood converges to a local maximum. A naive implementation of the clustering algorithm scales quadratically to the number of classes, since each time a word is moved between classes, all class bi-gram counts are potentially affected. However, by considering only those counts that actually change, the algorithm can be made to scale somewhere between linearly and quadratically to the number of classes (Martin et al., 1998).

In SMCM, partitions of each sentence are also determined. We used a Viterbi decoding (Deligne and BIMBOT, 1995) for the partition. We applied the exchange algorithm and the Viterbi decoding alternately until the log-likelihood converged to the local maximum.

Since the number of chunks is very large, for example, in our experiments we used about 3 million chunks, the computational cost is still large. We therefore employed the following two techniques. The first was to approximate the computation in the exchange algorithm; the second was to make use of bottom-up clustering to strengthen the convergence.

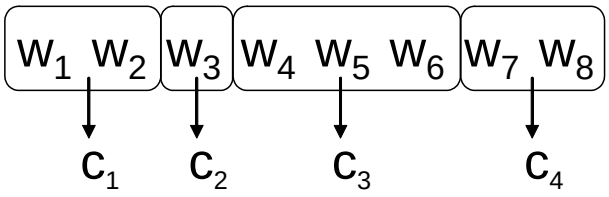


Figure 4: Example of assignment in semi-Markov class model. A sentence is partitioned into variable-length chunks and each chunk is assigned a unique class number.

In each step in the exchange algorithm, the approximate value of the change of the log-likelihood was examined, and the exchange algorithm applied only if the approximate value was larger than a predefined threshold.

The second technique was to reduce memory requirements. Since the matrices used in the exchange algorithm could become very large, we clustered chunks into 2 classes and then again we clustered these two into 2 each, thus obtaining 4 classes. This procedure was applied recursively until the number of classes reached a pre-defined number.

6 Experiments

6.1 Experimental Setup

We partitioned a BNC-corpus into *model-train*, *DLM-train-positive*, and *DLM-test-positive* sets. The numbers of sentences in *model-train*, *DLM-train-positive* and *DLM-test-positive* were 4500k, 250k, and 10k respectively. An NLM was built using *model-train* and Pseudo-Negative examples (250k sentences) were sampled from it. We mixed sentences from *DLM-train-positive* and the Pseudo-Negative examples and then shuffled the order of these sentences to make *DLM-train*. We also constructed *DLM-test* by mixing *DLM-test-positive* and 10k new (not already used) sentences from the Pseudo-Negative examples. We call the sentences from *DLM-train-positive* "positive" examples and the sentences from the Pseudo-Negative examples "negative" examples in the following. From these sentences the ones with less than 5 words were excluded beforehand because it was difficult to decide whether these sentences were correct or not (e.g.

	Accuracy (%)	Training time (s)
Linear classifier		
word tri-gram	51.28	137.1
POS tri-gram	52.64	85.0
SMCM bi-gram ($G = 100$)	51.79	304.9
SMCM bi-gram ($G = 500$)	54.45	422.1
3rd order Polynomial Kernel		
word tri-gram	73.65	20143.7
POS tri-gram	66.58	29622.9
SMCM bi-gram ($G = 100$)	67.11	37181.6
SMCM bi-gram ($G = 500$)	74.11	34474.7

Table 1: Performance on the evaluation data.

compound words).

Let G be the number of classes in SMCMs. Two SMCMs, one with $G = 100$ and the other with $G = 500$, were constructed from *model-train*. Each SMCM contained 2.8 million extracted chunks.

6.2 Experiments on Pseudo-Examples

We examined the property of a sentence being Pseudo-Negative, in order to justify our framework. A native English speaker and two non-native English speaker were asked to assign *correct*/*incorrect* labels to 100 sentences in *DLM-train*[1]. The result for an native English speaker was that all positive sentences were labeled as *correct* and all negative sentences except for one were labeled as *incorrect*. On the other hand, the results for non-native English speakers are 67% and 70%. From this result, we can say that the sampling method was able to generate incorrect sentences and if a classifier can discriminate them, the classifier can also discriminate between correct and incorrect sentences. Note that it takes an average of 25 seconds for the native English speaker to assign the label, which suggests that it is difficult even for a human to determine the correctness of a sentence.

We then examined whether it was possible to discriminate between correct and incorrect sentences using parsing methods, since if so, we could have used parsing as a classification tool. We examined 100 sentences using a phrase structure parser (Charniak and Johnson, 2005) and an HPSG parser (Miyao and Tsujii, 2005). All sentences were parsed correctly except for one positive example. This result indicates that correct sentences and pseudo-negative examples cannot be differentiated syntactically.

6.3 Experiments on DLM-PN

We investigated the performance of classifiers and the effect of different sets of features.

For N-grams and Part of Speech (POS), we used tri-gram features. For SMCM, we used bi-gram features. We used *DLM-train* as a training set. In all experiments, we set $C = 50.0$ where C is a parameter in the classification (Section 4). In all kernel experiments, a 3rd order polynomial kernel was used and values were computed using PKI (the inverted indexing method). Table 1 shows the accuracy results with different features, or in the case of the SMCMs, different numbers of classes. This result shows that the kernel method is important in achieving high performance. Note that the classifier with SMCM features performs as well as the one with *word*.

Table 2 shows the number of features in each method. Note that a new feature is added only if the classifier needs to update its parameters. These numbers are therefore smaller than the possible number of all candidate features. This result and the previous result indicate that SMCM achieves high performance with very few features.

We then examined the effect of PKI. Table 3 shows the results of the classifier with 3rd order polynomial kernel both with and without PKI. In this experiment, only $200K$ sentences in *DLM-train*

[1]Since the PLM also made use of the BNC-corpus for positive examples, we were not able to classify sentences based on word occurrences

	# of distinct features
word tri-gram	15773230
POS tri-gram	35376
SMCM ($G = 100$)	9335
SMCM ($G = 500$)	199745

Table 2: The number of features.

	training time (s)	prediction time (ms)
Baseline	37665.5	370.6
+ Index	4664.9	47.8

Table 3: Comparison between classification performance with/without index

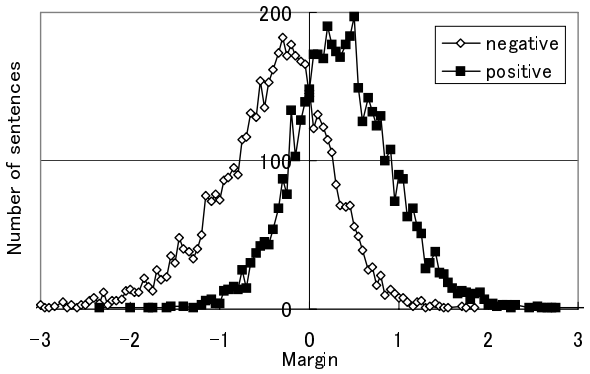


Figure 5: Margin distribution using SMCM bi-gram features.

were used for both experiments because training using all the training data would have required a much longer time than was possible with our experimental setup.

Figure 5 shows the margin distribution for positive and negative examples using SMCM bi-gram features. Although many examples are close to the border line (margin $= 0$), positive and negative examples are distributed on either side of 0. Therefore higher *recall* or *precision* could be achieved by using a pre-defined margin threshold other than 0.

Finally, we generated learning curves to examine the effect of the size of training data on performance. Figure 6 shows the result of the classification task using *SMCM*-bi-gram features. The result suggests that the performance could be further improved by enlarging the training data set.

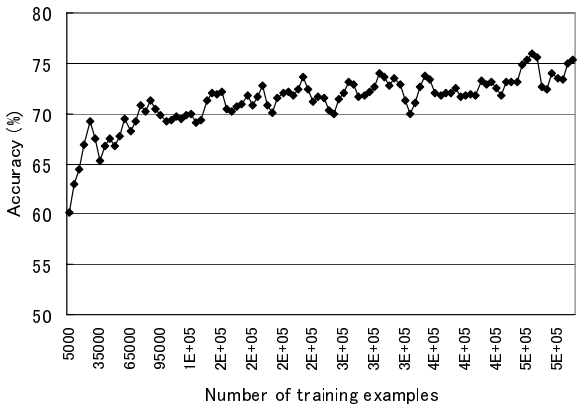


Figure 6: A learning curve for SMCM ($G = 500$). The accuracy is the percentage of sentences in the evaluation set classified correctly.

7 Discussion

Experimental results on pseudo-negative examples indicate that combination of features is effective in a sentence discrimination method. This could be because negative examples include many unsuitable combinations of words such as a sentence containing many nouns. Although in previous PLMs, combination of features has not been discussed except for the topic-based language model (David M. Blei, 2003; Wang et al., 2005), our result may encourage the study of the combination of features for language modeling.

A contrastive estimation method (Smith and Eisner, 2005) is similar to ours with regard to constructing pseudo-negative examples. They build a neighborhood of input examples to allow unsupervised estimation when, for example, a word is changed or deleted. A lattice is constructed, and then parameters are estimated efficiently. On the other hand, we construct independent pseudo-negative examples to enable training. Although the motivations of these studies are different, we could combine these two methods to discriminate sentences finely.

In our experiments, we did not examine the result of using other sampling methods, For example, it would be possible to sample sentences from a whole sentence maximum entropy model (Rosenfeld et al., 2001) and this is a topic for future research.

8 Conclusion

In this paper we have presented a novel discriminative language model using pseudo-negative examples. We also showed that an online margin-based learning method enabled us to use half a million sentences as training data and achieve 74% accuracy in the task of discrimination between correct and incorrect sentences. Experimental results indicate that while pseudo-negative examples can be seen as incorrect sentences, they are also close to correct sentences in that parsers cannot discriminate between them.

Our experimental results also showed that combination of features is important for discrimination between correct and incorrect sentences. This concept has not been discussed in previous probabilistic language models.

Our next step is to employ our model in machine translation and speech recognition. One main difficulty concerns how to encode global scores for the classifier in the local search space, and another is how to scale up the problem size in terms of the number of examples and features. We would like to see more refined online learning methods with kernels (Cheng et al., 2006; Dekel et al., 2005) that we could apply in these areas.

We are also interested in applications such as constructing an extended version of a spelling correction tool by identifying incorrect sentences.

Another interesting idea is to work with probabilistic language models directly without sampling and find ways to construct a more accurate discriminative model.

References

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. of ACL 05*, pages 173–180, June.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical report, Harvard Computer Science Technical report TR-10-98.

Li Cheng, S V N Vishwanathan, Dale Schuurmans, Shaojun Wang, and Terry Caelli. 2006. Implicit online learning with kernels. In *NIPS 2006*.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*.

Michael I. Jordan David M. Blei, Andrew Y. Ng. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research.*, 3:993–1022.

Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer. 2005. The forgetron: A kernel-based perceptron on a fixed budget. In *Proc. of NIPS*.

Sabine Deligne and Frédéric BIMBOT. 1995. Language modeling by variable length sequences: Theoretical formulation and evaluation of multigrams. In *Proc. ICASSP '95*, pages 169–172.

Jianfeng Gao, Hao Yu, Wei Yuan, and Peng Xu. 2005. Minimum sample risk methods for language modeling. In *Proc. of HLT/EMNLP*.

Taku Kudo and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. In *ACL*.

Sven Martin, Jörg Liermann, and Hermann Ney. 1998. Algorithms for bigram and trigram word clustering. *Speech Communicatoin*, 24(1):19–37.

Yusuke Miyao and Jun'ichi Tsujii. 2005. Probabilistic disambiguation models for wide-coverage hpsg parsing. In *Proc. of ACL 2005.*, pages 83–90, Ann Arbor, Michigan, June.

Brian Roark, Murat Saraclar, and Michael Collins. 2007. Discriminative n-gram language modeling. computer speech and language. *Computer Speech and Language*, 21(2):373–392.

Roni Rosenfeld, Stanley F. Chen, and Xiaojin Zhu. 2001. Whole-sentence exponential language models: a vehicle for linguistic-statistical integration. *Computers Speech and Language*, 15(1).

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. of ACL*.

John S. Taylor and Nello. Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambiridge Univsity Press.

Shaojun Wang, Shaomin Wang, Russell Greiner, Dale Schuurmans, and Li Cheng. 2005. Exploiting syntactic, semantic and lexical regularities in language modeling via directed markov random fields. In *Proc. of ICML*.

Detecting Erroneous Sentences using Automatically Mined Sequential Patterns

Guihua Sun *
Chongqing University
sunguihua5018@163.com

Xiaohua Liu **Gao Cong** **Ming Zhou**
Microsoft Research Asia
{xiaoliu, gaocong, mingzhou}@microsoft.com

Zhongyang Xiong
Chongqing University
zyxiong@cqu.edu.cn

John Lee †
MIT
jsylee@mit.edu

Chin-Yew Lin
Microsoft Research Asia
cyl@microsoft.com

Abstract

This paper studies the problem of identifying erroneous/correct sentences. The problem has important applications, e.g., providing feedback for writers of English as a Second Language, controlling the quality of parallel bilingual sentences mined from the Web, and evaluating machine translation results. In this paper, we propose a new approach to detecting erroneous sentences by integrating pattern discovery with supervised learning models. Experimental results show that our techniques are promising.

1 Introduction

Detecting erroneous/correct sentences has the following applications. First, it can provide feedback for writers of English as a Second Language (ESL) as to whether a sentence contains errors. Second, it can be applied to control the quality of parallel bilingual sentences mined from the Web, which are critical sources for a wide range of applications, such as statistical machine translation (Brown et al., 1993) and cross-lingual information retrieval (Nie et al., 1999). Third, it can be used to evaluate machine translation results. As demonstrated in (Corston-Oliver et al., 2001; Gamon et al., 2005), the better human reference translations can be distinguished from machine translations by a classification model, the worse the machine translation system is.

The previous work on identifying erroneous sentences mainly aims to find errors from the writing of ESL learners. The common mistakes (Yukio et al., 2001; Gui and Yang, 2003) made by ESL learners include spelling, lexical collocation, sentence structure, tense, agreement, verb formation, wrong Part-Of-Speech (POS), article usage, etc. The previous work focuses on grammar errors, including tense, agreement, verb formation, article usage, etc. However, little work has been done to detect sentence structure and lexical collocation errors.

Some methods of detecting erroneous sentences are based on manual rules. These methods (Heidorn, 2000; Michaud et al., 2000; Bender et al., 2004) have been shown to be effective in detecting certain kinds of grammatical errors in the writing of English learners. However, it could be expensive to write rules manually. Linguistic experts are needed to write rules of high quality; Also, it is difficult to produce and maintain a large number of non-conflicting rules to cover a wide range of grammatical errors. Moreover, ESL writers of different first-language backgrounds and skill levels may make different errors, and thus different sets of rules may be required. Worse still, it is hard to write rules for some grammatical errors, for example, detecting errors concerning the articles and singular plural usage (Nagata et al., 2006).

Instead of asking experts to write hand-crafted rules, statistical approaches (Chodorow and Leacock, 2000; Izumi et al., 2003; Brockett et al., 2006; Nagata et al., 2006) build statistical models to identify sentences containing errors. However, existing

*Work done while the author was a visiting student at MSRA

†Work done while the author was a visiting student at MSRA

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 81–88,
Prague, Czech Republic, June 2007.

statistical approaches focus on some pre-defined errors and the reported results are not attractive. Moreover, these approaches, e.g., (Izumi et al., 2003; Brockett et al., 2006) usually need errors to be specified and tagged in the training sentences, which requires expert help to be recruited and is time consuming and labor intensive.

Considering the limitations of the previous work, in this paper we propose a novel approach that is based on pattern discovery and supervised learning to successfully identify erroneous/correct sentences. The basic idea of our approach is to build a machine learning model to automatically classify each sentence into one of the two classes, "erroneous" and "correct." To build the learning model, we automatically extract *labeled sequential patterns* (LSPs) from both erroneous sentences and correct sentences, and use them as input features for classification models. Our main contributions are:

- We mine *labeled sequential patterns*(LSPs) from the preprocessed training data to build leaning models. Note that LSPs are also very different from N-gram language models that only consider continuous sequences.
- We also enrich the LSP features with other automatically computed linguistic features, including lexical collocation, language model, syntactic score, and function word density. In contrast with previous work focusing on (a specific type of) grammatical errors, our model can handle a wide range of errors, including grammar, sentence structure, and lexical choice.
- We empirically evaluate our methods on two datasets consisting of sentences written by Japanese and Chinese, respectively. Experimental results show that *labeled sequential patterns* are highly useful for the classification results, and greatly outperform other features. Our method outperforms Microsoft Word03 and ALEK (Chodorow and Leacock, 2000) from Educational Testing Service (ETS) in some cases. We also apply our learning model to machine translation (MT) data as a complementary measure to evaluate MT results.

The rest of this paper is organized as follows. The next section discusses related work. Section 3 presents the proposed technique. We evaluate our proposed technique in Section 4. Section 5 concludes this paper and discusses future work.

2 Related Work

Research on detecting erroneous sentences can be classified into two categories. The first category makes use of hand-crafted rules, e.g., template rules (Heidorn, 2000) and mal-rules in context-free grammars (Michaud et al., 2000; Bender et al., 2004). As discussed in Section 1, manual rule based methods have some shortcomings.

The second category uses statistical techniques to detect erroneous sentences. An unsupervised method (Chodorow and Leacock, 2000) is employed to detect grammatical errors by inferring negative evidence from TOEFL administrated by ETS. The method (Izumi et al., 2003) aims to detect omission-type and replacement-type errors and transformation-based leaning is employed in (Shi and Zhou, 2005) to learn rules to detect errors for speech recognition outputs. They also require specifying error tags that can tell the specific errors and their corrections in the training corpus. The phrasal Statistical Machine Translation (SMT) technique is employed to identify and correct writing errors (Brockett et al., 2006). This method must collect a large number of parallel corpora (pairs of erroneous sentences and their corrections) and performance depends on SMT techniques that are not yet mature. The work in (Nagata et al., 2006) focuses on a type of error, namely mass vs. count nouns. In contrast to existing statistical methods, our technique needs neither errors tagged nor parallel corpora, and is not limited to a specific type of grammatical error.

There are also studies on automatic essay scoring at document-level. For example, E-rater (Burstein et al., 1998), developed by the ETS, and Intelligent Essay Assessor (Foltz et al., 1999). The evaluation criteria for documents are different from those for sentences. A document is evaluated mainly by its organization, topic, diversity of vocabulary, and grammar while a sentence is done by grammar, sentence structure, and lexical choice.

Another related work is Machine Translation (MT) evaluation. Classification models are employed in (Corston-Oliver et al., 2001; Gamon et al., 2005)

to evaluate the well-formedness of machine translation outputs. The writers of ESL and MT normally make different mistakes: in general, ESL writers can write overall grammatically correct sentences with some local mistakes while MT outputs normally produce locally well-formed phrases with overall grammatically wrong sentences. Hence, the manual features designed for MT evaluation are not applicable to detect erroneous sentences from ESL learners.

LSPs differ from the traditional sequential patterns, e.g., (Agrawal and Srikant, 1995; Pei et al., 2001) in that LSPs are attached with class labels and we prefer those with discriminating ability to build classification model. In our other work (Sun et al., 2007), labeled sequential patterns, together with labeled tree patterns, are used to build pattern-based classifier to detect erroneous sentences. The classification method in (Sun et al., 2007) is different from those used in this paper. Moreover, instead of labeled sequential patterns, in (Sun et al., 2007) the most significant k labeled sequential patterns with constraints for each training sentence are mined to build classifiers. Another related work is (Jindal and Liu, 2006), where sequential patterns with labels are used to identify comparative sentences.

3 Proposed Technique

This section first gives our problem statement and then presents our proposed technique to build learning models.

3.1 Problem Statement

In this paper we study the problem of identifying erroneous/correct sentences. A set of training data containing correct and erroneous sentences is given. Unlike some previous work, our technique requires neither that the erroneous sentences are tagged with detailed errors, nor that the training data consist of parallel pairs of sentences (an error sentence and its correction). The erroneous sentence contains a wide range of errors on grammar, sentence structure, and lexical choice. We do not consider spelling errors in this paper.

We address the problem by building classification models. The main challenge is to automatically extract representative features for both correct and erroneous sentences to build effective classification models. We illustrate the challenge with an example. Consider an erroneous sentence, "*If* Maggie *will* go to supermarket, she *will* buy a bag for you." It is difficult for previous methods using statistical techniques to capture such an error. For example, N-gram language model is considered to be effective in writing evaluation (Burstein et al., 1998; Corston-Oliver et al., 2001). However, it becomes very expensive if $\mathrm{N} > 3$ and N-grams only consider continuous sequence of words, which is unable to detect the above error *"if...will...will"*.

We propose *labeled sequential patterns* to effectively characterize the features of correct and erroneous sentences (Section 3.2), and design some complementary features (Section 3.3).

3.2 Mining *Labeled Sequential Patterns* (LSP)

Labeled Sequential Patterns (LSP). A labeled sequential pattern, p, is in the form of $\texttt{LHS} \rightarrow \texttt{c}$, where LHS is a sequence and c is a class label. Let I be a set of items and L be a set of class labels. Let D be a sequence database in which each tuple is composed of a list of items in I and a class label in L. We say that a sequence $s_1 =< a_1, ..., a_m >$ is *contained* in a sequence $s_2 =< b_1, ..., b_n >$ if there exist integers $i_1, ...i_m$ such that $1 \leq i_1 < i_2 < ... < i_m \leq n$ and $a_j = b_{i_j}$ for all $j \in 1, ..., m$. Similarly, we say that a LSP p_1 is contained by p_2 if the sequence p_1.*LHS* is contained by p_2.*LHS* and $p_1.c = p_2.c$. Note that it is not required that s_1 appears continuously in s_2. We will further refine the definition of "contain" by imposing some constraints (to be explained soon). A LSP p is attached with two measures, *support* and *confidence*. The support of p, denoted by $\texttt{sup}(p)$, is the percentage of tuples in database D that contain the LSP p. The probability of the LSP p being true is referred to as "the confidence of p ", denoted by $\texttt{conf}(p)$, and is computed as $\frac{\texttt{sup}(p)}{\texttt{sup}(p.LHS)}$. The support is to measure the generality of the pattern p and minimum confidence is a statement of predictive ability of p.

Example 1: Consider a sequence database containing three tuples $t_1 = (< a, d, e, f >, E)$, $t_2 = (< a, f, e, f >, E)$ and $t_3 = (< d, a, f >, C)$. One example LSP $p_1 = < a, e, f > \rightarrow E$, which is contained in tuples t_1 and t_2. Its support is 66.7% and its confidence is 100%. As another example, LSP p_2

$= < a, f > \rightarrow E$ with support 66.7% and confidence 66.7%. p_1 is a better indication of class E than p_2. □

Generating Sequence Database. We generate the database by applying Part-Of-Speech (POS) tagger to tag each training sentence while keeping function words[1] and time words[2]. After the processing, each sentence together with its label becomes a database tuple. The function words and POS tags play important roles in both grammars and sentence structures. In addition, the time words are key clues in detecting errors of tense usage. The combination of them allows us to capture representative features for correct/erroneous sentences by mining LSPs. Some example LSPs include "$<$*a, NNS*$> \rightarrow Error$"(singular determiner preceding plural noun), and "$<$*yesterday, is*$> \rightarrow Error$". Note that the confidences of these LSPs are not necessary 100%.

First, we use MXPOST-Maximum Entropy Part of Speech Tagger Toolkit[3] for POS tags. The MXPOST tagger can provide fine-grained tag information. For example, noun can be tagged with "NN"(singular noun) and "NNS"(plural noun); verb can be tagged with "VB", "VBG", "VBN", "VBP", "VBD" and "VBZ". Second, the function words and time words that we use form a key word list. If a word in a training sentence is not contained in the key word list, then the word will be replaced by its POS. The processed sentence consists of POS and the words of key word list. For example, after the processing, the sentence "*In the past, John was kind to his sister*" is converted into "*In the past, NNP was JJ to his NN*", where the words "*in*", "*the*", "*was*", "*to*" and "*his*" are function words, the word "*past*" is time word, and "*NNP*", "*JJ*", and "*NN*" are POS tags.

Mining LSPs. The length of the discovered LSPs is flexible and they can be composed of contiguous or distant words/tags. Existing frequent sequential pattern mining algorithms (e.g. (Pei et al., 2001)) use *minimum support* threshold to mine frequent sequential patterns whose support is larger than the threshold. These algorithms are not sufficient for our problem of mining LSPs. In order to ensure that all our discovered LSPs are discriminating and are capable of predicting correct or erroneous sentences, we impose another constraint *minimum confidence*. Recall that the higher the confidence of a pattern is, the better it can distinguish between correct sentences and erroneous sentences. In our experiments, we empirically set *minimum support* at 0.1% and minimum confidence at 75%.

Mining LSPs is nontrivial since its search space is exponential, althought there have been a host of algorithms for mining frequent sequential patterns. We adapt the frequent sequence mining algorithm in (Pei et al., 2001) for mining LSPs with constraints.

Converting LSPs to Features. Each discovered LSP forms a binary feature as the input for classification model. If a sentence includes a LSP, the corresponding feature is set at 1.

The LSPs can characterize the correct/erroneous sentence structure and grammar. We give some examples of the discovered LSPs. (1) LSPs for erroneous sentences. For example, "$<$*this, NNS*$>$"(e.g. contained in "*this books* is stolen."), "$<$*past, is*$>$"(e.g. contained in "in the *past*, John *is* kind to his sister."), "$<$*one, of, NN*$>$"(e.g. contained in "it is *one of* important working *language*", "$<$*although, but*$>$"(e.g. contained in "*although* he likes it, *but* he can't buy it."), and "$<$*only, if, I, am*$>$"(e.g. contained in "*only if* my teacher has given permission, *I am* allowed to enter this room"). (2) LSPs for correct sentences. For instance, "$<$*would, VB*$>$"(e.g. contained in "he *would buy* it."), and "$<$*VBD, yeserday*$>$"(e.g. contained in "I *bought* this book *yesterday*.").

3.3 Other Linguistic Features

We use some linguistic features that can be computed automatically as complementary features.

Lexical Collocation (LC) Lexical collocation error (Yukio et al., 2001; Gui and Yang, 2003) is common in the writing of ESL learners, such as "*strong tea*" but not "*powerful tea.*" Our LSP features cannot capture all LCs since we replace some words with POS tags in mining LSPs. We collect five types of collocations: verb-object, adjective-noun, verb-adverb, subject-verb, and preposition-object from a general English corpus[4]. Correct LCs are collected

[1] http://www.marlodge.supanet.com/museum/funcword.html
[2] http://www.wjh.harvard.edu/%7Einquirer/Time%40.html
[3] http://www.cogsci.ed.ac.uk/∼jamesc/taggers/MXPOST.html
[4] The general English corpus consists of about 4.4 million native sentences.

by extracting collocations of high frequency from the general English corpus. Erroneous LC candidates are generated by replacing the word in correct collocations with its confusion words, obtained from WordNet, including synonyms and words with similar spelling or pronunciation. Experts are consulted to see if a candidate is a true erroneous collocation.

We compute three statistical features for each sentence below. (1) The first feature is computed by $\sum_{i=1}^{m} p(co_i)/n$, where m is the number of CLs, n is the number of collocations in each sentence, and probability $p(co_i)$ of each CL co_i is calculated using the method (Lü and Zhou, 2004). (2) The second feature is computed by the ratio of the number of unknown collocations (neither correct LCs nor erroneous LCs) to the number of collocations in each sentence. (3) The last feature is computed by the ratio of the number of erroneous LCs to the number of collocations in each sentence.

Perplexity from Language Model (PLM) Perplexity measures are extracted from a trigram language model trained on a general English corpus using the SRILM-SRI Language Modeling Toolkit (Stolcke, 2002). We calculate two values for each sentence: lexicalized trigram perplexity and part of speech (POS) trigram perplexity. The erroneous sentences would have higher perplexity.

Syntactic Score (SC) Some erroneous sentences often contain words and concepts that are locally correct but cannot form coherent sentences (Liu and Gildea, 2005). To measure the coherence of sentences, we use a statistical parser Toolkit (Collins, 1997) to assign each sentence a parser's score that is the related log probability of parsing. We assume that erroneous sentences with undesirable sentence structures are more likely to receive lower scores.

Function Word Density (FWD) We consider the density of function words (Corston-Oliver et al., 2001), i.e. the ratio of function words to content words. This is inspired by the work (Corston-Oliver et al., 2001) showing that function word density can be effective in distinguishing between human references and machine outputs. In this paper, we calculate the densities of seven kinds of function words [5] respectively as 7 features.

Dataset	Type	Source	Number
JC	(+)	the Japan Times newspaper and Model English Essay	16,857
	(-)	HEL (Hiroshima English Learners' Corpus) and JLE (Japanese Learners of English Corpus)	17,301
CC	(+)	the 21st Century newspaper	3,200
	(-)	CLEC (Chinese Learner Error Corpus)	3,199

Table 1: Corpora ((+): correct; (-): erroneous)

4 Experimental Evaluation

We evaluated the performance of our techniques with support vector machine (SVM) and Naive Bayesian (NB) classification models. We also compared the effectiveness of various features. In addition, we compared our technique with two other methods of checking errors, Microsoft Word03 and ALEK method (Chodorow and Leacock, 2000). Finally, we also applied our technique to evaluate the Machine Translation outputs.

4.1 Experimental Setup

Classification Models. We used two classification models, SVM[6] and NB classification model.

Data. We collected two datasets from different domains, Japanese Corpus (JC) and Chinese Corpus (CC). Table 1 gives the details of our corpora. In the learner's corpora, all of the sentences are erroneous. Note that our data does not consist of parallel pairs of sentences (one error sentence and its correction). The erroneous sentences includes grammar, sentence structure and lexical choice errors, but not spelling errors.

For each sentence, we generated five kinds of features as presented in Section 3. For a non-binary feature X, its value x is normalized by z-score, $norm(x) = \frac{x - mean(X)}{\sqrt{var(X)}}$, where `mean`(x) is the empirical mean of X and `var`(X) is the variance of X. Thus each sentence is represented by a vector.

Metrics We calculated the `precision`, `recall`, and `F-score` for correct and erroneous sentences, respectively, and also report the overall accuracy.

[5] including determiners/quantifiers, all pronouns, different pronoun types: Wh, 1^{st}, 2^{nd}, and 3^{rd} person pronouns, prepositions and adverbs, auxiliary verbs, and conjunctions.

[6] http://svmlight.joachims.org/

All the experimental results are obtained thorough 10-fold cross-validation.

4.2 Experimental Results

The Effectiveness of Various Features. The experiment is to evaluate the contribution of each feature to the classification. The results of SVM are given in Table 2. We can see that the performance of *labeled sequential patterns* (LSP) feature consistently outperforms those of all the other individual features. It also performs better even if we use all the other features together. This is because other features only provide some relatively abstract and simple linguistic information, whereas the discovered LSPs characterize significant linguistic features as discussed before. We also found that the results of NB are a little worse than those of SVM. However, all the features perform consistently on the two classification models and we can observe the same trend. Due to space limitation, we do not give results of NB.

In addition, the discovered LSPs themselves are intuitive and meaningful since they are intuitive features that can distinguish correct sentences from erroneous sentences. We discovered 6309 LSPs in JC data and 3742 LSPs in CC data. Some example LSPs discovered from erroneous sentences are *<a, NNS>* (*support*:0.39%, *confidence*:85.71%), *<to, VBD>* (*support*:0.11%, *confidence*:84.21%), and *<the, more, the, JJ>* (*support*:0.19%, *confidence*:0.93%) [7]; Similarly, we also give some example LSPs mined from correct sentences: *<NN, VBZ>* (*support*:2.29%, *confidence*:75.23%), and *<have, VBN, since>* (*support*:0.11%, *confidence*:85.71%) [8]. However, other features are abstract and it is hard to derive some intuitive knowledge from the opaque statistical values of these features.

As shown in Table 2, our technique achieves the highest accuracy, e.g. 81.75% on the Japanese dataset, when we use all the features. However, we also notice that the improvement is not very significant compared with using LSP feature individually (e.g. 79.63% on the Japanese dataset). The similar results are observed when we combined the features *PLM, SC, FWD,* and LC. This could be explained by two reasons: (1) A sentence may contain several kinds of errors. A sentence detected to be erroneous by one feature may also be detected by another feature; and (2) Various features give conflicting results. The two aspects suggest the directions of our future efforts to improve the performance of our models.

Comparing with Other Methods. It is difficult to find benchmark methods to compare with our technique because, as discussed in Section 2, existing methods often require error tagged corpora or parallel corpora, or focus on a specific type of errors. In this paper, we compare our technique with the grammar checker of Microsoft Word03 and the ALEK (Chodorow and Leacock, 2000) method used by ETS. ALEK is used to detect inappropriate usage of specific vocabulary words. Note that we do not consider spelling errors. Due to space limitation, we only report the `precision`, `recall`, `F-score` for erroneous sentences, and the overall accuracy.

As can be seen from Table 3, our method outperforms the other two methods in terms of overall accuracy, F-score, and recall, while the three methods achieve comparable precision. We realize that the grammar checker of Word is a general tool and the performance of ALEK (Chodorow and Leacock, 2000) can be improved if larger training data is used. We found that Word and ALEK usually cannot find sentence structure and lexical collocation errors, e.g., "*The more you listen to English, the easy it becomes*." contains the discovered LSP *<the, more, the, JJ>* $\rightarrow$ *Error*.

Cross-domain Results. To study the performance of our method on cross-domain data from writers of the same first-language background, we collected two datasets from Japanese writers, one is composed of 694 parallel sentences (+:347, -:347), and the other 1,671 non-parallel sentences (+:795, -:876). The two datasets are used as test data while we use JC dataset for training. Note that the test sentences come from different domains from the JC data. The results are given in the first two rows of Table 4. This experiment shows that our leaning model trained for one domain can be effectively applied to independent data in the other domains from the writes of the same first-language background, no matter whether the test data is parallel or not. We also noticed that

[7] a + plural noun; to + past tense format; the more + the + base form of adjective

[8] singular or mass noun + the 3^{rd} person singular present format; have + past participle format + since

Dataset	Feature	A	(-)F	(-)R	(-)P	(+)F	(+)R	(+)P
JC	LSP	79.63	80.65	85.56	76.29	78.49	73.79	83.85
	LC	69.55	71.72	77.87	66.47	67.02	61.36	73.82
	PLM	61.60	55.46	50.81	64.91	62	70.28	58.43
	SC	53.66	57.29	68.40	56.12	34.18	39.04	32.22
	FWD	68.01	72.82	86.37	62.95	61.14	49.94	78.82
	$LC + PLM + SC + FWD$	71.64	73.52	79.38	68.46	69.48	64.03	75.94
	$LSP + LC + PLM + SC + FWD$	81.75	81.60	81.46	81.74	81.90	82.04	81.76
CC	LSP	78.19	76.40	70.64	83.20	79.71	85.72	74.50
	LC	63.82	62.36	60.12	64.77	65.17	67.49	63.01
	PLM	55.46	64.41	80.72	53.61	40.41	30.22	61.30
	SC	50.52	62.58	87.31	50.64	13.75	14.33	13.22
	FWD	61.36	60.80	60.70	60.90	61.90	61.99	61.80
	$LC + PLM + SC + FWD$	67.69	67.62	67.51	67.77	67.74	67.87	67.64
	$LSP + LC + PLM + SC + FWD$	79.81	78.33	72.76	84.84	81.10	86.92	76.02

Table 2: The Experimental Results (A: overall accuracy; (-): erroneous sentences; (+): correct sentences; F: F-score; R: recall; P: precision)

Dataset	Model	A	(-)F	(-)R	(-)P
JC	Ours	81.39	81.25	81.24	81.28
	Word	58.87	33.67	21.03	84.73
	ALEK	54.69	20.33	11.67	78.95
CC	Ours	79.14	77.81	73.17	83.09
	Word	58.47	32.02	19.81	84.22
	ALEK	55.21	22.83	13.42	76.36

Table 3: The Comparison Results

Dataset	A	(-)F	(-)R	(-)P
JC(Train)+nonparallel(Test)	72.49	68.55	57.51	84.84
JC(Train)+parallel(Test)	71.33	69.53	65.42	74.18
JC + CC	79.98	79.72	79.24	80.23
JC(Train)+ CC(Test)	55.62	41.71	31.32	62.40
CC(Train)+ JC(Test)	57.57	23.64	16.94	39.11

Table 4: The Cross-domain Results of our Method

LSPs play dominating role in achieving the results. Due to space limitation, no details are reported.

To further see the performance of our method on data written by writers with different first-language backgrounds, we conducted two experiments. (1) We merge the JC dataset and CC dataset. The 10-fold cross-validation results on the merged dataset are given in the third row of Table 4. The results demonstrate that our models work well when the training data and test data contain sentences from different first-language backgrounds. (2) We use the JC dataset (resp. CC dataset) for training while the CC dataset (resp. JC dataset) is used as test data. As shown in the fourth (resp. fifth) row of Table 4, the results are worse than their corresponding results of Word given in Table 3. The reason is that the mistakes made by Japanese and Chinese are different, thus the learning model trained on one data does not work well on the other data. Note that our method is not designed to work in this scenario.

Application to Machine Translation Evaluation. Our learning models could be used to evaluate the MT results as an complementary measure. This is based on the assumption that if the MT results can be accurately distinguished from human references by our technique, the MT results are not natural and may contain errors as well.

The experiment was conducted using 10-fold cross validation on two LDC data, low-ranked and high-ranked data[9]. The results using SVM as classification model are given in Table 5. As expected, the classification accuracy on low-ranked data is higher than that on high-ranked data since low-ranked MT results are more different from human references than high-ranked MT results. We also found that LSPs are the most effective features. In addition, our discovered LSPs could indicate the common errors made by the MT systems and provide some suggestions for improving machine translation results.

As a summary, the mined LSPs are indeed effective for the classification models and our proposed technique is effective.

5 Conclusions and Future Work

This paper proposed a new approach to identifying erroneous/correct sentences. Empirical evaluating using diverse data demonstrated the effectiveness of

[9]One LDC data contains 14,604 low ranked (score 1-3) machine translations and the corresponding human references; the other LDC data contains 808 high ranked (score 3-5) machine translations and the corresponding human references

Data	Feature	A	(-)F	(-)R	(-)P	(+)F	(+)R	(+)P
Low-ranked data (1-3 score)	LSP	84.20	83.95	82.19	85.82	84.44	86.25	82.73
	LSP+LC+PLM+SC+FWD	86.60	86.84	88.96	84.83	86.35	84.27	88.56
High-ranked data (3-5 score)	LSP	71.74	73.01	79.56	67.59	70.23	64.47	77.40
	LSP+LC+PLM+SC+FWD	72.87	73.68	68.95	69.20	71.92	67.22	77.60

Table 5: The Results on Machine Translation Data

our techniques. Moreover, we proposed to mine LSPs as the input of classification models from a set of data containing correct and erroneous sentences. The LSPs were shown to be much more effective than the other linguistic features although the other features were also beneficial.

We will investigate the following problems in the future: (1) to make use of the discovered LSPs to provide detailed feedback for ESL learners, e.g. the errors in a sentence and suggested corrections; (2) to integrate the features effectively to achieve better results; (3) to further investigate the application of our techniques for MT evaluation.

References

Rakesh Agrawal and Ramakrishnan Srikant. 1995. Mining sequential patterns. In *ICDE*.

Emily M. Bender, Dan Flickinger, Stephan Oepen, Annemarie Walsh, and Timothy Baldwin. 2004. Arboretum: Using a precision grammar for grammmar checking in call. In *Proc. InSTIL/ICALL Symposium on Computer Assisted Learning*.

Chris Brockett, William Dolan, and Michael Gamon. 2006. Correcting esl errors using phrasal smt techniques. In *ACL*.

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

Jill Burstein, Karen Kukich, Susanne Wolff, Chi Lu, Martin Chodorow, Lisa Braden-Harder, and Mary Dee Harris. 1998. Automated scoring using a hybrid feature identification technique. In *Proc. ACL*.

Martin Chodorow and Claudia Leacock. 2000. An unsupervised method for detecting grammatical errors. In *NAACL*.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proc. ACL*.

Simon Corston-Oliver, Michael Gamon, and Chris Brockett. 2001. A machine learning approach to the automatic evaluation of machine translation. In *Proc. ACL*.

P.W. Foltz, D. Laham, and T.K. Landauer. 1999. Automated essay scoring: Application to educational technology. In *EdMedia '99*.

Michael Gamon, Anthony Aue, and Martine Smets. 2005. Sentence-level mt evaluation without reference translations: Beyond language modeling. In *Proc. EAMT*.

Shicun Gui and Huizhong Yang. 2003. *Zhongguo Xuexizhe Yingyu Yuliaohu. (Chinese Learner English Corpus)*. Shanghai: Shanghai Waiyu Jiaoyu Chubanshe. (In Chinese).

George E. Heidorn. 2000. *Intelligent Writing Assistance*. Handbook of Natural Language Processing. Robert Dale, Hermann Moisi and Harold Somers (ed.). Marcel Dekker.

Emi Izumi, Kiyotaka Uchimoto, Toyomi Saiga, Thepchai Supnithi, and Hitoshi Isahara. 2003. Automatic error detection in the japanese learners' english spoken data. In *Proc. ACL*.

Nitin Jindal and Bing Liu. 2006. Identifying comparative sentences in text documents. In *SIGIR*.

Ding Liu and Daniel Gildea. 2005. Syntactic features for evaluation of machine translation. In *Proc. ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*.

Yajuan Lü and Ming Zhou. 2004. Collocation translation acquisition using monolingual corpora. In *Proc. ACL*.

Lisa N. Michaud, Kathleen F. McCoy, and Christopher A. Pennington. 2000. An intelligent tutoring system for deaf learners of written english. In *Proc. 4th International ACM Conference on Assistive Technologies*.

Ryo Nagata, Atsuo Kawai, Koichiro Morihiro, and Naoki Isu. 2006. A feedback-augmented method for detecting errors in the writing of learners of english. In *Proc. ACL*.

Jian-Yun Nie, Michel Simard, Pierre Isabelle, and Richard Durand. 1999. Cross-language information retrieval based on parallel texts and automatic mining of parallel texts from the web. In *SIGIR*, pages 74–81.

Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, and Helen Pinto. 2001. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proc. ICDE*.

Yongmei Shi and Lina Zhou. 2005. Error detection using linguistic features. In *HLT/EMNLP*.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *Proc. ICSLP*.

Guihua Sun, Gao Cong, Xiaohua Liu, Chin-Yew Lin, and Ming Zhou. 2007. Mining sequential patterns and tree patterns to detect erroneous sentences. In *AAAI*.

Tono Yukio, T. Kaneko, H. Isahara, T. Saiga, and E. Izumi. 2001. The standard speaking test corpus: A 1 million-word spoken corpus of japanese learners of english and its implications for l2 lexicography. In *ASIALEX: Asian Bilingualism and the Dictionary*.

Vocabulary Decomposition for Estonian Open Vocabulary Speech Recognition

Antti Puurula and Mikko Kurimo
Adaptive Informatics Research Centre
Helsinki University of Technology
P.O.Box 5400, FIN-02015 HUT, Finland
{puurula, mikkok}@cis.hut.fi

Abstract

Speech recognition in many morphologically rich languages suffers from a very high out-of-vocabulary (OOV) ratio. Earlier work has shown that vocabulary decomposition methods can practically solve this problem for a subset of these languages. This paper compares various vocabulary decomposition approaches to open vocabulary speech recognition, using Estonian speech recognition as a benchmark. Comparisons are performed utilizing large models of 60000 lexical items and smaller vocabularies of 5000 items. A large vocabulary model based on a manually constructed morphological tagger is shown to give the lowest word error rate, while the unsupervised morphology discovery method Morfessor Baseline gives marginally weaker results. Only the Morfessor-based approach is shown to adequately scale to smaller vocabulary sizes.

1 Introduction

1.1 OOV problem

Open vocabulary speech recognition refers to automatic speech recognition (ASR) of continuous speech, or "speech-to-text" of spoken language, where the recognizer is expected to recognize any word spoken in that language. This capability is a recent development in ASR, and is required or beneficial in many of the current applications of ASR technology. Moreover, large vocabulary speech recognition is not possible in most languages of the world without first developing the tools needed for open vocabulary speech recognition. This is due to a fundamental obstacle in current ASR called the out-of-vocabulary (OOV) problem.

The OOV problem refers to the existence of words encountered that a speech recognizer is unable to recognize, as they are not covered in the vocabulary. The OOV problem is caused by three intertwined issues. Firstly, the language model training data and the test data always come from different samplings of the language, and the mismatch between test and training data introduces some OOV words, the amount depending on the difference between the data sets. Secondly, ASR systems always use finite and preferably small sized vocabularies, since the speed of decoding rapidly slows down as the vocabulary size is increased. Vocabulary sizes depend on the application domain, sizes larger than 60000 being very rare. As some of the words encountered in the training data are left out of the vocabulary, there will be OOV words during recognition. The third and final issue is the fundamental one; languages form novel sentences not only by combining words, but also by combining sub-word items called morphs to make up the words themselves. These morphs in turn correspond to abstract grammatical items called morphemes, and morphs of the same morpheme are called allomorphs of that morpheme. The study of these facets of language is aptly called morphology, and has been largely neglected in modern ASR technology. This is due to

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 89–95,
Prague, Czech Republic, June 2007.

ASR having been developed primarily for English, where the OOV problem is not as severe as in other languages of the world.

1.2 Relevance of morphology for ASR

Morphologies in natural languages are characterized typologically using two parameters, called indexes of synthesis and fusion. Index of synthesis has been loosely defined as the ratio of morphs per word forms in the language(Comrie, 1989), while index of fusion refers to the ratio of morphs per morpheme. High frequency of verb paradigms such as "hear, hear + d, hear + d" would result in a high synthesis, low fusion language, whereas high frequency of paradigms such as "sing, sang, sung" would result in almost the opposite. Counting distinct item types and not instances of the types, the first example would have 2 word forms, 2 morphs and 2 morphemes, the second 3 word forms, 3 morphs and 1 morpheme. Although in the first example, there are 3 word instances of the 2 word forms, the latter word form being an ambiguous one referring to two distinct grammatical constructions. It should also be noted that the first morph of the first example has 2 pronunciations. Pronunciational boundaries do not always follow morphological ones, and a morph may and will have several pronunciations that depend on context, if the language in question has significant orthographic irregularity.

As can be seen, both types of morphological complexity increase the amount of distinct word forms, resulting in an increase in the OOV rate of any finite sized vocabulary for that language. In practice, the OOV increase caused by synthesis is much larger, as languages can have thousands of different word forms per word that are caused by addition of processes of word formation followed by inflections. Thus the OOV problem in ASR has been most pronounced in languages with much synthesis, regardless of the amount of fusion. The morpheme-based modeling approaches evaluated in this work are primarily intended for fixing the problem caused by synthesis, and should work less well or even adversely when attempted with low synthesis, high fusion languages. It should be noted that models based on finite state transducers have been shown to be adequate for describing fusion as well(Koskenniemi, 1983), and further work should evaluate these types of models in ASR of languages with higher indexes of fusion.

1.3 Approaches for solving the OOV problem

The traditional method for reducing OOV would be to simply increase the vocabulary size so that the rate of OOV words becomes sufficiently low. Naturally this method fails when the words are derived, compounded or inflected forms of rarer words. While this approach might still be practical in languages with a low index of synthesis such as English, it fails with most languages in the world. For example, in English with language models (LM) of 60k words trained from the Gigaword Corpus V.2(Graff et al., 2005), and testing on a very similar Voice of America -portion of TDT4 speech corpora(Kong and Graff, 2005), this gives a OOV rate of 1.5%. It should be noted that every OOV causes roughly two errors in recognition, and vocabulary decomposition approaches such as the ones evaluated here give some benefits to word error rate (WER) even in recognizing languages such as English(Bisani and Ney, 2005).

Four different approaches to lexical unit selection are evaluated in this work, all of which have been presented previously. These are hence called "word", "hybrid", "morph" and "grammar". The word approach is the default approach to lexical item selection, and is provided here as a baseline for the alternative approaches. The alternatives tested here are all based on decomposing the in-vocabulary words, OOV words, or both, in LM training data into sequences of sub-word fragments. During recognition the decoder can then construct the OOV words encountered as combinations of these fragments. Word boundaries are marked in LMs with tokens so that the words can be reconstructed from the sub-word fragments after decoding simply by removing spaces between fragments, and changing the word boundaries tokens to spaces. As splitting to sub-word items makes the span of LM histories shorter, higher order n-grams must be used to correct this. Varigrams(Siivola and Pellom, 2005) are used in this work, and to make LMs trained with each approach comparable, the varigrams have been grown to roughly sizes of 5 million counts. It should be noted that the names for the approaches here are somewhat arbitrary, as from a theoretical perspec-

tive both morph- and grammar-based approaches try to model the grammatical morph set of a language, difference being that "morph" does this with an unsupervised data-driven machine learning algorithm, whereas "grammar" does this using segmentations from a manually constructed rule-based morphological tagger.

2 Modeling approaches

2.1 Word approach

The first approach evaluated in this work is the traditional word based LM, where items are simply the most frequent words in the language model training data. OOV words are simply treated as unknown words in language model training. This has been the default approach to selection of lexical items in speech recognition for several decades, and as it has been sufficient in English ASR, there has been limited interest in any alternatives.

2.2 Hybrid approach

The second approach is a recent refinement of the traditional word-based approach. This is similar to what was introduced as "flat hybrid model"(Bisani and Ney, 2005), and it tries to model OOV-words as sequences of words and fragments. "Hybrid" refers to the LM histories being composed of hybrids of words and fragments, while "flat" refers to the model being composed of one n-gram model instead of several models for the different item types. The models tested in this work differ in that since Estonian has a very regular phonemic orthography, grapheme sequences can be directly used instead of more complex pronunciation modeling. Subsequently the fragments used are just one grapheme in length.

2.3 Morph approach

The morph-based approach has shown superior results to word-based models in languages of high synthesis and low fusion, including Estonian. This approach, called "Morfessor Baseline" is described in detail in (Creutz et al., 2007). An unsupervised machine learning algorithm is used to discover the morph set of the language in question, using minimum description length (MDL) as an optimization criterion. The algorithm is given a word list of the language, usually pruned to about 100 000 words, that it proceeds to recursively split to smaller items, using gains in MDL to optimize the item set. The resulting set of morphs models the morph set well in languages of high synthesis, but as it does not take fusion into account any manner, it should not work in languages of high fusion. It neither preserves information about pronunciations, and as these do not follow morph boundaries, the approach is unsuitable in its basic form to languages of high orthographic irregularity.

2.4 Grammar approach

The final approach applies a manually constructed rule-based morphological tagger(Alumäe, 2006). This approach is expected to give the best results, as the tagger should give the ideal segmentation along the grammatical morphs that the unsupervised and language-independent morph approach tries to find. To make this approach more comparable to the morph models, OOV morphs are modeled as sequences of graphemes similar to the hybrid approach. Small changes to the original approach were also made to make the model comparable to the other models presented here, such as using the tagger segmentations as such and not using pseudo-morphemes, as well as not tagging the items in any manner. This approach suffers from the same handicaps as the morph approach, as well as from some additional ones: morphological analyzers are not readily available for most languages, they must be tailored by linguists for new datasets, and it is an open problem as to how pronunciation dictionaries should be written for grammatical morphs in languages with significant orthographic irregularity.

2.5 Text segmentation and language modeling

For training the LMs, a subset of 43 million words from the Estonian Segakorpus was used(Segakorpus, 2005), preprocessed with a morphological analyzer(Alumäe, 2006). After selecting the item types, segmenting the training corpora and generation of a pronunciation dictionary, LMs were trained for each lexical item type. Table 1 shows the text format for LM training data after segmentation with each model. As can be seen, the word-based approach doesn't use word boundary tokens. To keep the LMs comparable between each model-

model	text segmentation
word 5k	voodis reeglina loeme
word 60k	voodis reeglina loeme
hybrid 5k	v o o d i s <w> reeglina <w> l o e m e
hybrid 60k	voodis <w> reeglina <w> loeme
morph 5k	voodi s <w> re e g lina <w> loe me
morph 60k	voodi s <w> reegli na <w> loe me
grammar 5k	voodi s <w> reegli na <w> loe me
grammar 60k	voodi s <w> reegli na <w> loe me

Table 1. Sample segmented texts for each model.

ing approach, growing varigram models(Siivola and Pellom, 2005) were used with no limits as to the order of n-grams, but limiting the number of counts to 4.8 and 5 million counts. In some models this growing method resulted in the inclusion of very frequent long item sequences to the varigram, up to a 28-gram. Models of both 5000 and 60000 lexical items were trained in order to test if and how the modeling approaches would scale to smaller and therefore much faster vocabularies. Distribution of counts in n-gram orders can be seen in figure 1.

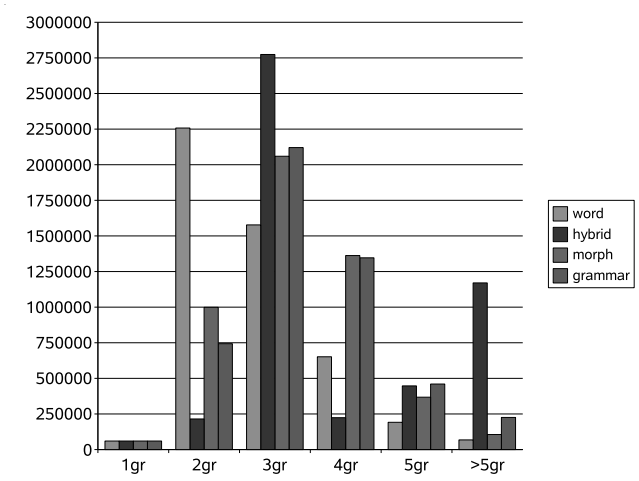


Figure 1. Number of counts included for each n-gram order in the 60k varigram models.

The performance of the statistical language models is often evaluated by perplexity or cross-entropy. However, we decided to only report the real ASR performance, because perplexity does not suit well to the comparison of models that use different lexica, have different OOV rates and have lexical units of different lengths.

3 Experimental setup

3.1 Evaluation set

Acoustic models for Estonian ASR were trained on the Estonian Speechdat-like corpus(Meister et al., 2002). This consists of spoken newspaper sentences and shorter utterances, read over a telephone by 1332 different speakers. The data therefore was quite clearly articulated, but suffered from 8kHz sample rate, different microphones, channel noises and occasional background noises. On top of this the speakers were selected to give a very broad coverage of different dialectal varieties of Estonian and were of different age groups. For these reasons, in spite of consisting of relatively common word forms from newspaper sentences, the database can be considered challenging for ASR.

Held-out sentences were from the same corpus used as development and evaluation set. 8 different sentences from 50 speakers each were used for evaluation, while sentences from 15 speakers were used for development. LM scaling factor was optimized for each model separately on the development set. On total over 200 hours of data from the database was used for acoustic model training, of which less than half was speech.

3.2 Decoding

The acoustic models were Hidden Markov Models (HMM) with Gaussian Mixture Models (GMM) for state modeling based on 39-dimensional MFCC+P+D+DD features, with windowed cepstral mean subtraction (CMS) of 1.25 second window. Maximum likelihood linear transformation (MLLT) was used during training. State-tied cross-word triphones and 3 left-to-right states were used, state durations were modeled using gamma distributions. On total 3103 tied states and 16 Gaussians per state were used.

Decoding was done with the decoder developed at TKK(Pylkkönen, 2005), which is based on a one-pass Viterbi beam search with token passing on a lexical prefix tree. The lexical prefix tree included a cross-word network for modeling triphone contexts, and the nodes in the prefix tree were tied at the triphone state level. Bigram look-ahead models were

used in speeding up decoding, in addition to pruning with global beam, history, histogram and word end pruning. Due to the properties of the decoder and varigram models, very high order n-grams could be used without significant degradation in decoding speed.

As the decoder was run with only one pass, adaptation was not used in this work. In preliminary experiments simple adaptation with just constrained maximum likelihood linear regression (CMLLR) was shown to give as much as 20 % relative word error rate reductions (RWERR) with this dataset. Adaptation was not used, since it interacts with the model types, as well as with the WER from the first round of decoding, providing larger RWERR for the better models. With high WER models, adaptation matrices are less accurate, and it is also probable that the decomposition methods yield more accurate matrices, as they produce results where fewer HMM-states are misrecognized. These issues should be investigated in future research.

After decoding, the results were post-processed by removing words that seemed to be sequences of junk fragments: consonant-only sequences and 1-phoneme words. This treatment should give very significant improvements with noisy data, but in preliminary experiments it was noted that the use of sentence boundaries resulted in almost 10% RWERR weaker results for the approaches using fragments, as that almost negates the gains achieved from this post-processing. Since sentence boundary forcing is done prior to junk removal, it seems to work erroneously when it is forced to operate on noisy data. Sentence boundaries were nevertheless used, as in the same experiments the word-based models gained significantly from their use, most likely because they cannot use the fragment items for detection of acoustic junk, as the models with fragments can.

4 Results

Results of the experiments were consistent with earlier findings(Hirsimäki et al., 2006; Kurimo et al., 2006). Traditional word based LMs showed the worst performance, with all of the recently proposed alternatives giving better results. Hybrid LMs consistently outperformed traditional word-based LMs in both large and small vocabulary conditions. The two morphology-driven approaches gave similar and clearly superior results. Only the morph approach seems to scale down well to smaller vocabulary sizes, as the WER for the grammar approach increased rapidly as size of the vocabulary was decreased.

size	word	hybrid	morph	grammar
60000	53.1	47.1	39.4	38.7
5000	82.0	63.0	43.5	47.6

Table 2. Word error rates for the models (WER %).

Table 2 shows the WER for the large (60000) and small (5000) vocabulary sizes and different modeling approaches. Table 3 shows the corresponding letter error rates (LER). LERs are more comparable across some languages than WERs, as WER depends more on factors such as length, morphological complexity, and OOV of the words. However, for within-language and between-model comparisons, the RWERR should still be a valid metric, and is also usable in languages that do not use a phonemic writing system. The RWERRs of different novel methods seems to be comparable between different languages as well. Both WER and LER are high considering the task. However, standard methods such as adaptation were not used, as the intention was only to study the RWERR of the different approaches.

size	word	hybrid	morph	grammar
60000	17.8	15.8	12.4	12.3
5000	35.5	20.8	14.4	15.4

Table 3. Letter error rates for the models (LER %).

5 Discussion

Four different approaches to lexical item selection for large and open vocabulary ASR in Estonian were evaluated. It was shown that the three approaches utilizing vocabulary decomposition give substantial improvements over the traditional word based approach, and make large vocabulary ASR technology possible for languages similar to Estonian, where the traditional approach fails due to very

high OOV rates. These include memetic relatives Finnish and Turkish, among other languages that have morphologies of high fusion, low synthesis and low orthographic irregularity.

5.1 Performance of the approaches

The morpheme-based approaches outperformed the word- and hybrid-based approaches clearly. The results for "hybrid" are in in the range suggested by earlier work(Bisani and Ney, 2005). One possible explanation for the discrepancy between the hybrid and morpheme-based approaches would be that the morpheme-based approaches capture items that make sense in n-gram modeling, as morphs are items that the system of language naturally operates on. These items would then be of more use when trying to predict unseen data(Creutz et al., 2007). As modeling pronunciations is much more straightforward in Estonian, the morpheme-based approaches do not suffer from erroneous pronunciations, resulting in clearly superior performance.

As for the superiority of the "grammar" over the unsupervised "morph", the difference is marginal in terms of RWERR. The grammatical tagger was tailored by hand for that particular language, whereas Morfessor method is meant to be unsupervised and language independent. There are further arguments that would suggest that the unsupervised approach is one that should be followed; only "morph" scaled well to smaller vocabulary sizes, the usual practice of pruning the word list to produce smaller morph sets gives better results than here and most importantly, it is questionable if "grammar" can be taken to languages with high indexes of fusion and orthographic irregularity, as the models have to take these into account as well.

5.2 Comparison to previous results

There are few previous results published on Estonian open vocabulary ASR. In (Alumäe, 2006) a WER of 44.5% was obtained with word-based trigrams and a WER of 37.2% with items similar to ones from "grammar" using the same speech corpus as in this work. Compared to the present work, the WER for the morpheme-based models was measured with compound words split in both hypothesis and reference texts, making the task slightly easier than here. In (Kurimo et al., 2006) a WER of 57.6% was achieved with word-based varigrams and a WER of 49.0% with morphs-based ones. This used the same evaluation set as this work, but had slightly different LMs and different acoustic modelling which is the main reason for the higher WER levels. In summary, morpheme-based approaches seem to consistently outperform the traditional word based one in Estonian ASR, regardless of the specifics of the recognition system, test set and models.

In (Hirsimäki et al., 2006) a corresponding comparison of unsupervised and grammar-based morphs was presented in Finnish, and the grammar-based model gave a significantly higher WER in one of the tasks. This result is interesting, and may stem from a number of factors, among them the different decoder and acoustic models, 4-grams versus varigrams, as well as differences in post-processing. Most likely the difference is due to lack of coverage for domain-specific words in the Finnish tagger, as it has a 4.2% OOV rate on the training data. On top of this the OOV words are modeled simply as grapheme sequences, instead of modeling only OOV morphs in that manner, as is done in this work.

5.3 Open problems in vocabulary decomposition

As stated in the introduction, modeling languages with high indexes of fusion such as Arabic will require more complex vocabulary decomposition approaches. This is verified by recent empirical results, where gains obtained from simple morphological decomposition seem to be marginal(Kirchhoff et al., 2006; Creutz et al., 2007). These languages would possibly need novel LM inference algorithms and decoder architectures. Current research seems to be heading in this direction, with weighted finite state transducers becoming standard representations for the vocabulary instead of the lexical prefix tree.

Another issue in vocabulary decomposition is orthographic irregularity, as the items resulting from decomposition do not necessarily have unambiguous pronunciations. As most modern recognizers use the Viterbi approximation with vocabularies of one pronunciation per item, this is problematic. One solution to this is expanding the different items with tags according to pronunciation, shifting the problem to language modeling(Creutz et al., 2007). For example, English plural "s" would expand to "s#1"

with pronunciation "/s/", and "s#2" with pronunciation "/z/", and so on. In this case the vocabulary size increases by the amount of different pronunciations added. The new items will have pronunciations that depend on their language model context, enabling the prediction of pronunciations with language model probabilities. The only downside to this is complicating the search for optimal vocabulary decomposition, as the items should make sense in both pronunciational and morphological terms.

One can consider the originally presented hybrid approach as an approach to vocabulary decomposition that tries to keep the pronunciations of the items as good as possible, whereas the morph approach tries to find items that make sense in terms of morphology. This is obviously due to the methods having been developed on very different types of languages. The morph approach was developed for the needs of Finnish speech recognition, which is a high synthesis, moderate fusion and very low orthographic irregularity language, whereas the hybrid approach in (Bisani and Ney, 2005) was developed for English, which has low synthesis, moderate fusion, and very high orthographic irregularity. A universal approach to vocabulary decomposition would have to take all of these factors into account.

Acknowledgements

The authors would like to thank Dr. Tanel Alumäe from Tallinn University of Technology for help in performing experiments with Estonian speech and text databases. This work was supported by the Academy of Finland in the project: *New adaptive and learning methods in speech recognition.*

References

Bernard Comrie. 1972. *Language Universals and Linguistic Typology*, Second Edition. Athenæum Press Ltd, Gateshead, UK.

Kimmo Koskenniemi. 1983. *Two-level Morphology: a General Computational Model for Word-Form Recognition and Production*. University of Helsinki, Helsinki, Finland.

Tanel Alumäe. 2006. Methods for Estonian Large Vocabulary Speech Recognition. *PhD Thesis.* Tallinn University of Technology. Tallinn, Estonia.

Maximilian Bisani, Hermann Ney. 2005. Open Vocabulary Speech Recognition with Flat Hybrid Models. *INTERSPEECH-2005*, 725–728.

Janne Pylkkönen. 2005. An Efficient One-pass Decoder for Finnish Large Vocabulary Continuous Speech Recognition. *Proceedings of The 2nd Baltic Conference on Human Language Technologies*, 167–172. HLT'2005. Tallinn, Estonia.

Vesa Siivola, Bryan L. Pellom. 2005. Growing an n-Gram Language Model. *INTERSPEECH-2005*, 1309–1312.

David Graff, Junbo Kong, Ke Chen and Kazuaki Maeda. 2005. LDC Gigaword Corpora: English Gigaword Second Edition. In *LDC link: http://www.ldc.upenn.edu/Catalog/index.jsp*.

Junbo Kong and David Graff. 2005. TDT4 Multilingual Broadcast News Speech Corpus. In *LDC link: http://www.ldc.upenn.edu/Catalog/index.jsp*.

Segakorpus. 2005. Segakorpus - Mixed Corpus of Estonian. Tartu University. *http://test.cl.ut.ee/korpused/*.

Einar Meister, Jürgen Lasn and Lya Meister 2002. Estonian SpeechDat: a project in progress. In *Proceedings of the Fonetiikan Päivät - Phonetics Symposium 2002 in Finland*, 21–26.

Katrin Kirchhoff, Dimitra Vergyri, Jeff Bilmes, Kevin Duh and Andreas Stolcke 2006. Morphology-based language modeling for conversational Arabic speech recognition. *Computer Speech & Language* 20(4):589–608.

Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraclar and Andreas Stolcke 2007. Analysis of Morph-Based Speech Recognition and the Modeling of Out-of-Vocabulary Words Across Languages To appear in *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics*. NAACL-HLT 2007, Rochester, NY, USA

Mikko Kurimo, Antti Puurula, Ebru Arisoy, Vesa Siivola, Teemu Hirsimäki, Janne Pylkkönen, Tanel Alumae and Murat Saraclar 2006. Unlimited vocabulary speech recognition for agglutinative languages. In *Human Language Technology, Conference of the North American Chapter of the Association for Computational Linguistics*. HLT-NAACL 2006. New York, USA

Teemu Hirsimäki, Mathias Creutz, Vesa Siivola, Mikko Kurimo, Sami Virpioja and Janne Pylkkönen 2006. Unlimited vocabulary speech recognition with morph language models applied to Finnish. *Computer Speech & Language* 20(4):515–541.

Phonological Constraints and Morphological Preprocessing for Grapheme-to-Phoneme Conversion

Vera Demberg
School of Informatics
University of Edinburgh
Edinburgh, EH8 9LW, GB
v.demberg@sms.ed.ac.uk

Helmut Schmid
IMS
University of Stuttgart
D-70174 Stuttgart
schmid@ims.uni-stuttgart.de

Gregor Möhler
Speech Technologies
IBM Deutschland Entwicklung
D-71072 Böblingen
moehler@de.ibm.com

Abstract

Grapheme-to-phoneme conversion (g2p) is a core component of any text-to-speech system. We show that adding simple syllabification and stress assignment constraints, namely 'one nucleus per syllable' and 'one main stress per word', to a joint n-gram model for g2p conversion leads to a dramatic improvement in conversion accuracy.

Secondly, we assessed morphological preprocessing for g2p conversion. While morphological information has been incorporated in some past systems, its contribution has never been quantitatively assessed for German. We compare the relevance of morphological preprocessing with respect to the morphological segmentation method, training set size, the g2p conversion algorithm, and two languages, English and German.

1 Introduction

Grapheme-to-Phoneme conversion (g2p) is the task of converting a word from its spelling (e.g. "Sternanisöl", Engl: star-anise oil) to its pronunciation (/ˈʃtɛrnʔaniːsʔøːl/). Speech synthesis modules with a g2p component are used in text-to-speech (TTS) systems and can be be applied in spoken dialogue systems or speech-to-speech translation systems.

1.1 Syllabification and Stress in g2p conversion

In order to correctly synthesize a word, it is not only necessary to convert the letters into phonemes, but also to syllabify the word and to assign word stress.

The problems of word phonemization, syllabification and word stress assignment are inter-dependent. Information about the position of a syllable boundary helps grapheme-to-phoneme conversion. (Marchand and Damper, 2005) report a word error rate (WER) reduction of approx. 5 percentage points for English when the letter string is augmented with syllabification information. The same holds vice-versa: we found that WER was reduced by 50% when running our syllabifier on phonemes instead of letters (see Table 4). Finally, word stress is usually defined on syllables; in languages where word stress is assumed[1] to partly depend on syllable weight (such as German or Dutch), it is important to know where exactly the syllable boundaries are in order to correctly calculate syllable weight. For German, (Müller, 2001) show that information about stress assignment and the position of a syllable within a word improve g2p conversion.

1.2 Morphological Preprocessing

It has been argued that using morphological information is important for languages where morphology has an important influence on pronunciation, syllabification and word stress such as German, Dutch, Swedish or, to a smaller extent, also English (Sproat, 1996; Möbius, 2001; Pounder and Kommenda, 1986; Black et al., 1998; Taylor, 2005). Unfortunately, these papers do not quantify the contribution of morphological preprocessing in the task.

Important questions when considering the integration of a morphological component into a speech

[1] This issue is controversial among linguists; for an overview see (Jessen, 1998).

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 96–103,
Prague, Czech Republic, June 2007.

synthesis system are 1) How large are the improvements to be gained from morphological preprocessing? 2) Must the morphological system be perfect or can performance improvements also be reached with relatively simple morphological components? and 3) How much does the benefit to be expected from explicit morphological information depend on the g2p algorithm? To determine these factors, we compared morphological segmentations based on manual morphological annotation from CELEX to two rule-based systems and several unsupervised data-based approaches. We also analysed the role of explicit morphological preprocessing on data sets of different sizes and compared its relevance with respect to a decision tree and a joint n-gram model for g2p conversion.

The paper is structured as follows: We introduce the g2p conversion model we used in section 2 and explain how we implemented the phonological constraints in section 3. Section 4 is concerned with the relation between morphology, word pronunciation, syllabification and word stress in German, and presents different sources for morphological segmentation. In section 5, we evaluate the contribution of each of the components and compare our methods to state-of-the-art systems. Section 6 summarizes our results.

2 Methods

We used a joint n-gram model for the grapheme-to-phoneme conversion task. Models of this type have previously been shown to yield very good g2p conversion results (Bisani and Ney, 2002; Galescu and Allen, 2001; Chen, 2003). Models that do not use *joint* letter-phoneme states, and therefore are not conditional on the preceding letters, but only on the actual letter and the preceding phonemes, achieved inferior results. Examples of such approaches using Hidden Markov Models are (Rentzepopoulos and Kokkinakis, 1991) (who applied the HMM to the related task of phoneme-to-grapheme conversion), (Taylor, 2005) and (Minker, 1996).

The g2p task is formulated as searching for the most probable sequence of phonemes given the orthographic form of a word. One can think of it as a tagging problem where each letter is tagged with a (possibly empty) phoneme-sequence p. In our particular implementation, the model is defined as a higher-order Hidden Markov Model, where the hidden states are a letter–phoneme-sequence pair $\langle l; p\rangle$, and the observed symbols are the letters l. The output probability of a hidden state is then equal to one, since all hidden states that do not contain the observed letter are pruned directly.

The model for grapheme-to-phoneme conversion uses the Viterbi algorithm to efficiently compute the most probable sequence $\hat{p}_1^n$ of phonemes $\hat{p}_1, \hat{p}_2, ..., \hat{p}_n$ for a given letter sequence l_1^n. The probability of a letter–phon-seq pair depends on the *k* preceding letter–phon-seq pairs. Dummy states '#' are appended at both ends of each word to indicate the word boundary and to ensure that all conditional probabilities are well-defined.

$$\hat{p}_1^n = \arg\max_{p_1^n} \prod_{i=1}^{n+1} P(\langle l; p\rangle_i \,|\, \langle l; p\rangle_{i-k}^{i-1})$$

In an integrated model where g2p conversion, syllabification and word stress assignment are all performed at the same time, a state additionally contains a syllable boundary flag b and a stress flag a, yielding $\langle l; p; b; a\rangle_i$.

As an alternative architecture, we also designed a modular system that comprises one component for syllabification and one for word stress assignment. The model for syllabification computes the most probable sequence $\hat{b}_1^n$ of syllable boundary-tags $\hat{b}_1$, $\hat{b}_2, ..., \hat{b}_n$ for a given letter sequence l_1^n.

$$\hat{b}_1^n = \arg\max_{b_1^n} \prod_{i=1}^{n+1} P(\langle l; b\rangle_i \,|\, \langle l; b\rangle_{i-k}^{i-1})$$

The stress assignment model works on syllables. It computes the most probable sequence $\hat{a}_1^n$ of word accent-tags $\hat{a}_1, \hat{a}_2, ..., \hat{a}_n$ for a given syllable sequence syl_1^n.

$$\hat{a}_1^n = \arg\max_{a_1^n} \prod_{i=1}^{n+1} P(\langle syl; a\rangle_i \,|\, \langle syl; a\rangle_{i-k}^{i-1})$$

2.1 Smoothing

Because of major data sparseness problems, smoothing is an important issue, in particular for the stress model which is based on syllable–stress-tag pairs. Performance varied by up to 20% in function of the smoothing algorithm chosen. Best results were obtained when using a variant of Modified Kneser-Ney Smoothing[2] (Chen and Goodman, 1996).

[2]For a formal definition, see(Demberg, 2006).

2.2 Pruning

In the g2p-model, each letter can on average map onto one of 12 alternative phoneme-sequences. When working with 5-grams[3], there are about 12^5 = 250,000 state sequences. To improve time and space efficiency, we implemented a simple pruning strategy that only considers the t best states at any moment in time. With a threshold of $t = 15$, about 120 words are processed per minute on a 1.5GHz machine. Conversion quality is only marginally worse than when the whole search space is calculated.

Running time for English is faster, because the average number of candidate phonemes for each letter is lower. We measured running time (including training and the actual g2p conversion in 10-fold cross validation) for a Perl implementation of our algorithm on the English NetTalk corpus (20,008 words) on an Intel Pentium 4, 3.0 GHz machine. Running time was less than 1h for each of the following three test conditions: c1) g2p conversion only, c2) syllabification first, then g2p conversion, c3) simultaneous g2p conversion and syllabification, given perfect syllable boundary input, c4) simultaneous g2p conversion and syllabification when correct syllabification is not available beforehand. This is much faster than the times for Pronunciation by Analogy (PbA) (Marchand and Damper, 2005) on the same corpus. Marchand and Damper reported a processing time of several hours for c4), two days for c2) and several days for c3).

2.3 Alignment

Our current implementation of the joint n-gram model is not integrated with an automatic alignment procedure. We therefore first aligned letters and phonemes in a separate, semi-automatic step. Each letter was aligned with zero to two phonemes and, in the integrated model, zero or one syllable boundaries and stress markers.

3 Integration of Phonological Constraints

When analysing the results from the model that does g2p conversion, syllabification and stress assignment in a single step, we found that a large proportion of the errors was due to the violation of basic phonological constraints.

Some syllables had no syllable nucleus, while others contained several vowels. The reason for the errors is that German syllables can be very long and therefore sparse, often causing the model to back-off to smaller contexts. If the context is too small to cover the syllable, the model cannot decide whether the current syllable contains a nucleus.

In stress assignment, this problem is even worse: the context window rarely covers the whole word. The algorithm does not know whether it already assigned a word stress outside the context window. This leads to a high error rate with 15-20% of incorrectly stressed words. Thereof, 37% have more than one main stress, about 27% are not assigned any stress and 36% are stressed in the wrong position. This means that we can hope to reduce the errors by almost 2/3 by using phonological constraints.

Word stress assignment is a difficult problem in German because the underlying processes involve some deeper morphological knowledge which is not available to the simple model. In complex words, stress mainly depends on morphological structure (i.e. on the compositionality of compounds and on the stressing status of affixes). Word stress in simplex words is assumed to depend on the syllable position within the word stem and on syllable weight. The current language-independent approach does not model these processes, but only captures some of its statistics.

Simple constraints can help to overcome the problem of lacking context by explicitly requiring that every syllable must have exactly one syllable nucleus and that every word must have exactly one syllable receiving primary stress.

3.1 Implementation

Our goal is to find the most probable syllabified and stressed phonemization of a word that does not violate the constraints. We tried two different approaches to enforce the constraints.

In the first variant (v1), we modified the probability model to enforce the constraints. Each state now corresponds to a sequence of 4-tuples consisting of a letter l, a phoneme sequence p, a syllable boundary tag b, an accent tag a (as before) plus two

[3]There is a trade-off between long context windows which capture the context accurately and data sparseness issues. The optimal value k for the context window size depends on the source language (existence of multiletter graphemes, complexity of syllables etc.).

new flags A and N which indicate whether an accent/nucleus precedes or not. The A and N flags of the new state are a function of its accent and syllable boundary tag and the A and N flag of the preceding state. They split each state into four new states. The new transition probabilities are defined as:

$$P(\langle l; p; b; a\rangle_i \mid \langle l; p; b; a\rangle_{i-k}^{i-1}, A, N)$$

The probability is 0 if the transition violates a constraint, e.g., when the A flag is set and a_i indicates another accent.

A positive side effect of the syllable flag is that it stores separate phonemization probabilities for consonants in the syllable onset vs. consonants in the coda. The flag in the onset is 0 since the nucleus has not yet been encountered, whereas it is set to 1 in the coda. In German, this can e.g. help in for syllable-final devoicing of voiced stops and fricatives.

The increase in the number of states aggravates sparse-data problems. Therefore, we implemented another variant (v2) which uses the same set of states (with A and N flags), but with the transition probabilities of the original model, which did not enforce the constraints. Instead, we modified the Viterbi algorithm to eliminate the invalid transitions: For example, a transition from a state with the A flag set to a state where a_i introduces a second stress, is always ignored. On small data sets, better results were achieved with v2 (see Table 5).

4 Morphological Preprocessing

In German, information about morphological boundaries is needed to correctly insert glottal stops [ʔ] in complex words, to determine irregular pronunciation of affixes (*v* is pronounced [v] in *vertikal* but [f] in *ver+ticker+n*, and the suffix syllable *heit* is not stressed although superheavy and word final) and to disambiguate letters (e.g. *e* is always pronounced /ə/ when occurring in inflectional suffixes). Vowel length and quality has been argued to also depend on morphological structure (Pounder and Kommenda, 1986). Furthermore, morphological boundaries overrun default syllabification rules, such as the maximum onset principle.

Applying default syllabification to the word "Sternanisöl" would result in a syllabification into `Ster-na-ni-söl` (and subsequent phonemization to something like /ʃtɛʁˈnaːnizøːl/) instead of `Stern-a-nis-öl` (/ˈʃtɛʁnʔaniːsʔøːl/). Syllabification in turn affects phonemization since voiced fricatives and stops are devoiced in syllable-final position. Morphological information also helps for graphemic parsing of words such as "Röschen" (Engl: little rose) where the morphological boundary between *Rös* and *chen* causes the string *sch* to be transcribed to /sç/ instead of /ʃ/. Similar ambiguities can arise for all other sounds that are represented by several letters in orthography (e.g. doubled consonants, diphtongs, *ie*, *ph*, *th*), and is also valid for English. Finally, morphological information is also crucial to determine word stress in morphologically complex words.

4.1 Methods for Morphological Segmentation

Good segmentation performance on arbitrary words is hard to achieve. We compared several approaches with different amounts of built-in knowledge. The morphological information is encoded in the letter string, where different digits represent different kinds of morphological boundaries (prefixes, stems, derivational and inflectional suffixes).

Manual Annotation from CELEX

To determine the upper bound of what can be achieved when exploiting perfect morphological information, we extracted morphological boundaries and boundary types from the CELEX database.

The manual annotation is not perfect as it contains some errors and many cases where words are not decomposed entirely. The words tagged [F] for "lexicalized inflection", e.g. *gedrängt* (past participle of *drängen*, Engl: push) were decomposed semi-automatically for the purpose of this evaluation. As expected, annotating words with CELEX morphological segmentation yielded the best g2p conversion results. Manual annotation is only available for a small number of words. Therefore, only automatically annotated morphological information can scale up to real applications.

Rule-based Systems

The traditional approach is to use large morpheme lexica and a set of rules that segment words into affixes and stems. Drawbacks of using such a system are the high development costs, limited coverage

and problems with ambiguity resolution between alternative analyses of a word.

The two rule-based systems we evaluated, the ETI[4] morphological system and SMOR[5] (Schmid et al., 2004), are both high-quality systems with large lexica that have been developed over several years. Their performance results can help to estimate what can realistically be expected from an automatic segmentation system. Both of the rule-based systems achieved an F-score of approx. 80% morphological boundaries correct with respect to CELEX manual annotation.

Unsupervised Morphological Systems

Most attractive among automatic systems are methods that use unsupervised learning, because these require neither an expert linguist to build large rule-sets and lexica nor large manually annotated word lists, but only large amounts of tokenized text, which can be acquired e.g. from the internet. Unsupervised methods are in principle[6] language-independent, and can therefore easily be applied to other languages.

We compared four different state-of-the-art unsupervised systems for morphological decomposition (cf. (Demberg, 2006; Demberg, 2007)). The algorithms were trained on a German newspaper corpus (taz), containing about 240 million words. The same algorithms have previously been shown to help a speech recognition task (Kurimo et al., 2006).

5 Experimental Evaluations

5.1 Training Set and Test Set Design

The German corpus used in these experiments is CELEX (German Linguistic User Guide, 1995). CELEX contains a phonemic representation of each word, syllable boundaries and word stress information. Furthermore, it contains manually verified morphological boundaries.

Our training set contains approx. 240,000 words and the test set consists of 12,326 words. The test set is designed such that word stems in training and test sets are disjoint, i.e. the inflections of a certain stem are either all in the training set or all in the test set. Stem overlap between training and test set only occurs in compounds and derivations. If a simple random splitting (90% for training set, 10% for test set) is used on inflected corpora, results are much better: Word error rates (WER) are about 60% lower when the set of stems in training and test set are not disjoint. The same effect can also be observed for the syllabification task (see Table 4).

5.2 Results for the Joint n-gram Model

The joint n-gram model is language-independent. An aligned corpus with words and their pronunciations is needed, but no further adaptation is required.

Table 1 shows the performance of our model in comparison to alternative approaches on the German and English versions of the CELEX corpus, the English NetTalk corpus, the English Teacher's Word Book (TWB) corpus, the English beep corpus and the French Brulex corpus. The joint n-gram model performs significantly better than the decision tree (essentially based on (Lucassen and Mercer, 1984)), and achieves scores comparable to the Pronunciation by Analogy (PbA) algorithm (Marchand and Damper, 2005). For the Nettalk data, we also compared the influence of syllable boundary annotation from a) automatically learnt and b) manually annotated syllabification information on phoneme accuracy. Automatic syllabification for our model integrated phonological constraints (as described in section 3.1), and therefore led to an improvement in phoneme accuracy, while the word error rate increased for the PbA approach, which does not incorporate such constraints.

(Chen, 2003) also used a joint n-gram model. The two approaches differ in that Chen uses small chunks ($\langle\langle l : |0..1| \rangle : \langle p : |0..1| \rangle\rangle$ pairs only) and iteratively optimizes letter-phoneme alignment during training. Chen smoothes higher-order Markov Models with Gaussian Priors and implements additional language modelling such as consonant doubling.

[4] Eloquent Technology, Inc. (ETI) TTS system. http://www.mindspring.com/~ssshp/ssshp_cd/ss_eloq.htm

[5] The lexicon used by SMOR, IMSLEX, contains morphologically complex entries, which leads to high precision and low recall. The results reported here refer to a version of SMOR, where the lexicon entries were decomposed using a rather naïve high-recall segmentation method. SMOR itself does not disambiguate morphological analyses of a word. Our version used transition weights learnt from CELEX morphological annotation. For more details refer to (Demberg, 2006).

[6] Most systems make some assumptions about the underlying morphological system, for instance that morphology is a concatenative process, that stems have a certain minimal length or that prefixing and suffixing are the most relevant phenomena.

corpus	size	jnt n-gr	PbA	Chen	dec.tree
G - CELEX	230k	7.5%			15.0%
E - Nettalk	20k	35.4%	34.65%	34.6%	
a) auto.syll		35.3%	35.2%		
b) man.syll		29.4%	28.3%		
E - TWB	18k	28.5%	28.2%		
E - beep	200k	14.3%	13.3%		
E - CELEX	100k	23.7%			31.7%
F - Brulex	27k	10.9%			

Table 1: Word error rates for different g2p conversion algorithms. Constraints were only used in the E-Nettalk auto. syll condition.

5.3 Benefit of Integrating Constraints

The accuracy improvements achieved by integrating the constraints (see Table 2) are highly statistically significant. The numbers for conditions "G-syllab.+stress+g2p" and "E-syllab.+g2p" in Table 2 differ from the numbers for "G-CELEX" and "E-Nettalk" in Table 1 because phoneme conversion errors, syllabification errors and stress assignment errors are all counted towards word error rates reported in Table 2.

Word error rate in the combined g2p-syllable-stress model was reduced from 21.5% to 13.7%. For the separate tasks, we observed similar effects: The word error rate for inserting syllable boundaries was reduced from 3.48% to 3.1% on letters and from 1.84% to 1.53% on phonemes. Most significantly, word error rate was decreased from 30.9% to 9.9% for word stress assignment on graphemes.

We also found similarly important improvements when applying the syllabification constraint to English grapheme-to-phoneme conversion and syllabification. This suggests that our findings are not specific to German but that this kind of general constraints can be beneficial for a range of languages.

	no constr.	constraint(s)
G - syllab.+stress+g2p	21.5%	13.7%
G - syllab. on letters	3.5%	3.1%
G - syllab. on phonemes	1.84%	1.53%
G - stress assignm. on letters	30.9%	9.9%
E - syllab.+g2p	40.5%	37.5%
E - syllab. on phonemes	12.7%	8.8%

Table 2: Improving performance on g2p conversion, syllabification and stress assignment through the introduction of constraints. The table shows word error rates for German CELEX (G) and English NetTalk (E).

5.4 Modularity

Modularity is an advantage if the individual components are more specialized to their task (e.g. by applying a particular level of description of the problem, or by incorporating some additional source of knowledge).In a modular system, one component can easily be substituted by another – for example, if a better way of doing stress assignment in German was found. On the other hand, keeping everything in one module for strongly inter-dependent tasks (such as determining word stress and phonemization) allows us to simultaneously optimize for the best combination of phonemes and stress.

Best results were obtained from the joint n-gram model that does syllabification, stress assignment and g2p conversion all in a single step and integrates phonological constraints for syllabification and word stress (WER = 14.4% using method v1, WER = 13.7% using method v2). If the modular architecture is chosen, best results are obtained when g2p conversion is done before syllabification and stress assignment (15.2% WER), whereas doing syllabification and stress assignment first and then g2p conversion leads to a WER of 16.6%. We can conclude from this finding that an integrated approach is superior to a pipeline architecture for strongly inter-dependent tasks such as these.

5.5 The Contribution of Morphological Preprocessing

A statistically significant (according to a two-tailed t-test) improvement in g2p conversion accuracy (from 13.7% WER to 13.2% WER) was obtained with the manually annotated morphological boundaries from CELEX. The segmentation from both of the rule-based systems (ETI and SMOR) also resulted in an accuracy increase with respect to the baseline (13.6% WER), which is not annotated with morphological boundaries.

Among the unsupervised systems, best results[7] on the g2p task with morphological annotation were obtained with the RePortS system (Keshava and Pitler, 2006). But none of the segmentations led to an error reduction when compared to a baseline that used no morphological information (see Table 3). Word error rate even increased when the quality of the

[7]For all results refer to (Demberg, 2006).

	Precis.	Recall	F-Meas.	WER
RePortS (unsuperv.)	71.1%	50.7%	59.2%	15.1%
no morphology				13.7%
SMOR (rule-based)	87.1%	80.4%	83.6%	
ETI (rule-based)	75.4%	84.1%	79.5%	13.6%
CELEX (manual)	100%	100%	100%	13.2%

Table 3: Systems evaluation on German CELEX manual annotation and on the g2p task using a joint n-gram model. WERs refer to implementation v2.

morphological segmentation was too low (the unsupervised algorithms achieved 52%-62% F-measure with respect to CELEX manual annotation).

Table 4 shows that high-quality morphological information can also significantly improve performance on a syllabification task for German. We used the syllabifier described in (Schmid et al., 2005), which works similar to the joint n-gram model used for g2p conversion. Just as for g2p conversion, we found a significant accuracy improvement when using the manually annotated data, a smaller improvement for using data from the rule-based morphological system, and no improvement when using segmentations from an unsupervised algorithm. Syllabification works best when performed on phonemes, because syllables are phonological units and therefore can be determined most easily in terms of phonological entities such as phonemes.

Whether morphological segmentation is worth the effort depends on many factors such as training set size, the g2p algorithm and the language considered.

	disj. stems	random
RePortS (unsupervised morph.)	4.95%	
no morphology	3.10%	0.72%
ETI (rule-based morph.)	2.63%	
CELEX (manual annot.)	1.91%	0.53%
on phonemes	1.53%	0.18%

Table 4: Word error rates (WER) for syllabification with a joint n-gram model for two different training and test set designs (see Section 5.1).

Morphology for Data Sparseness Reduction

Probably the most important aspect of morphological segmentation information is that it can help to resolve data sparseness issues. Because of the additional knowledge given to the system through the morphological information, similarly-behaving letter sequences can be grouped more effectively.

Therefore, we hypothesized that morphological information is most beneficial in situations where the training corpus is rather small. Our findings confirm this expectation, as the relative error reduction through morphological annotation for a training corpus of 9,600 words is 6.67%, while it is only 3.65% for a 240,000-word training corpus.

In our implementation, the stress flags and syllable flags we use to enforce the phonological constraints increase data sparseness. We found v2 (the implementation that uses the states without stress and syllable flags and enforces the constraints by eliminating invalid transitions, cf. section 3.1) to outperform the integrated version, v1, and more significantly in the case of more severe data sparseness. The only condition when we found v1 to perform better than v2 was with a large data set and additional data sparseness reduction through morphological annotation, as in section 4 (see Table 5).

WER: designs	v1		v2	
data set size	240k	9.6k	240k	9.6k
no morph.	14.4%	32.3%	**13.7%**	25.5%
CELEX	**12.5%**	29%	13.2%	23.8%

Table 5: The interactions of constraints in training and different levels of data sparseness.

g2p Conversion Algorithms

The benefit of using morphological preprocessing is also affected by the algorithm that is used for g2p conversion. Therefore, we also evaluated the relative improvement of morphological annotation when using a decision tree for g2p conversion.

Decision trees were one of the first data-based approaches to g2p and are still widely used (Kienappel and Kneser, 2001; Black et al., 1998). The tree's efficiency and ability for generalization largely depends on pruning and the choice of possible questions. In our implementation, the decision tree can ask about letters within a context window of five back and five ahead, about five phonemes back and groups of letters (e.g. consonants vs. vowels).

Both the decision tree and the joint n-gram model convert graphemes to phonemes, insert syllable boundaries and assign word stress in a single step (marked as "WER-ss" in Table 6. The implementation of the joint n-gram model incorporates the phonological constraints described in section 3 ("WER-ss+"). Our main finding is that the joint n-gram model profits less from morphological annotation. Without the constraints, the performance

difference is smaller: the joint n-gram model then achieves a word error rate of 21.5% on the no-morphology-condition.

In very recent work, (Demberg, 2007) developed an unsupervised algorithm (f-meas: 68%; an extension of RePortS) whose segmentations improve g2p when using a the decision tree (PER: 3.45%).

	decision tree		joint n-gram	
	PER	WER-ss	PER	WER-ss$^+$
RePortS	3.83%	28.3%		15.1%
no morph.	3.63%	26.59%	2.52%	13.7%
ETI	2.8%	21.13%	2.53%	13.6%
CELEX	2.64%	21.64%	2.36%	13.2%

Table 6: The effect of morphological preprocessing on phoneme error rates (PER) and word error rates (WER) in grapheme-to-phoneme conversion.

Morphology for other Languages

We also investigated the effect of morphological information on g2p conversion and syllabification in English, using manually annotated morphological boundaries from CELEX and the automatic unsupervised RePortS system which achieves an F-score of about 77% for English. The cases where morphological information affects word pronunciation are relatively few in comparison to German, therefore the overall effect is rather weak and we did not even find improvements with perfect boundaries.

6 Conclusions

Our results confirm that the integration of phonological constraints 'one nucleus per syllable' and 'one main stress per word' can significantly boost accuracy for g2p conversion in German and English. We implemented the constraints using a joint n-gram model for g2p conversion, which is language-independent and well-suited to the g2p task.

We systematically evaluated the benefit to be gained from morphological preprocessing on g2p conversion and syllabification. We found that morphological segmentations from rule-based systems led to some improvement. But the magnitude of the accuracy improvement strongly depends on the g2p algorithm and on training set size. State-of-the-art unsupervised morphological systems do not yet yield sufficiently good segmentations to help the task, if a good conversion algorithm is used: Low quality segmentation even led to higher error rates.

Acknowledgments

We would like to thank Hinrich Schütze, Frank Keller and the ACL reviewers for valuable comments and discussion.
The first author was supported by Evangelisches Studienwerk e.V. Villigst.

References

M. Bisani and H. Ney. 2002. Investigations on joint multigram models for grapheme-to-phoneme conversion. In *ICSLP*.

A. Black, K. Lenzo, and V. Pagel. 1998. Issues in building general letter to sound rules. In *3. ESCA on Speech Synthesis*.

SF Chen and J Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proc. of ACL*.

S. F. Chen. 2003. Conditional and joint models for grapheme-to-phoneme conversion. In *Eurospeech*.

V. Demberg. 2006. Letter-to-phoneme conversion for a German TTS-System. Master's thesis. *IMS, Univ. of Stuttgart*.

V. Demberg. 2007. A language-independent unsupervised model for morphological segmentation. In *Proc. of ACL-07*.

L. Galescu and J. Allen. 2001. Bi-directional conversion between graphemes and phonemes using a joint n-gram model. In *Proc. of the 4th ISCA Workshop on Speech Synthesis*.

CELEX German Linguistic User Guide, 1995. *Center for Lexical Information*. Max-Planck-Institut for Psycholinguistics, Nijmegen.

M. Jessen, 1998. *Word Prosodic Systems in the Languages of Europe*. Mouton de Gruyter: Berlin.

S. Keshava and E. Pitler. 2006. A simpler, intuitive approach to morpheme induction. In *Proceedings of 2nd Pascal Challenges Workshop*, pages 31–35, Venice, Italy.

A. K. Kienappel and R. Kneser. 2001. Designing very compact decision trees for grapheme-to-phoneme transcription. In *Eurospeech*, Scandinavia.

M. Kurimo, M. Creutz, M. Varjokallio, E. Arisoy, and M. Saraclar. 2006. Unsupervsied segmentation of words into morphemes – Challenge 2005: An introduction and evaluation report. In *Proc. of 2nd Pascal Challenges Workshop*, Italy.

J. Lucassen and R. Mercer. 1984. An information theoretic approach to the automatic determination of phonemic baseforms. In *ICASSP 9*.

Y. Marchand and R. I. Damper. 2005. Can syllabification improve pronunciation by analogy of English? *Natural Language Engineering*.

W. Minker. 1996. Grapheme-to-phoneme conversion - an approach based on hidden markov models.

B. Möbius. 2001. *German and Multilingual Speech Synthesis*. phonetic AIMS, Arbeitspapiere des Instituts für Maschinelle Spachverarbeitung.

K. Müller. 2001. Automatic detection of syllable boundaries combining the advantages of treebank and bracketed corpora training. In *Proceedings of ACL*, pages 402–409.

A. Pounder and M. Kommenda. 1986. Morphological analysis for a German text-to-speech system. In *COLING 1986*.

P.A. Rentzepopoulos and G.K. Kokkinakis. 1991. Phoneme to grapheme conversion using HMM. In *Eurospeech*.

H. Schmid, A. Fitschen, and U. Heid. 2004. SMOR: A German computational morphology covering derivation, composition and inflection. In *Proc. of LREC*.

H. Schmid, B. Möbius, and J. Weidenkaff. 2005. Tagging syllable boundaries with hidden Markov models. IMS, unpub.

R. Sproat. 1996. Multilingual text analysis for text-to-speech synthesis. In *Proc. ICSLP '96*, Philadelphia, PA.

P. Taylor. 2005. Hidden Markov models for grapheme to phoneme conversion. In *INTERSPEECH*.

Redundancy Ratio: An Invariant Property of the Consonant Inventories of the World's Languages

Animesh Mukherjee, Monojit Choudhury, Anupam Basu, Niloy Ganguly
Department of Computer Science and Engineering,
Indian Institute of Technology, Kharagpur
{animeshm,monojit,anupam,niloy}@cse.iitkgp.ernet.in

Abstract

In this paper, we put forward an information theoretic definition of the *redundancy* that is observed across the sound inventories of the world's languages. Through rigorous statistical analysis, we find that this redundancy is an invariant property of the consonant inventories. The statistical analysis further unfolds that the vowel inventories do not exhibit any such property, which in turn points to the fact that the organizing principles of the vowel and the consonant inventories are quite different in nature.

1 Introduction

Redundancy is a strikingly common phenomenon that is observed across many natural systems. This redundancy is present mainly to reduce the risk of the complete loss of information that might occur due to accidental errors (Krakauer and Plotkin, 2002). Moreover, redundancy is found in every level of granularity of a system. For instance, in biological systems we find redundancy in the codons (Lesk, 2002), in the genes (Woollard, 2005) and as well in the proteins (Gatlin, 1974). A linguistic system is also not an exception. There is for example, a number of words with the same meaning (synonyms) in almost every language of the world. Similarly, the basic unit of language, the human speech sounds or the phonemes, is also expected to exhibit some sort of a redundancy in the information that it encodes.

In this work, we attempt to mathematically capture the redundancy observed across the sound (more specifically the consonant) inventories of the world's languages. For this purpose, we present an information theoretic definition of redundancy, which is calculated based on the set of *features*[1] (Trubetzkoy, 1931) that are used to express the consonants. An interesting observation is that this quantitative feature-based measure of redundancy is *almost* an invariance over the consonant inventories of the world's languages. The observation is important since it can shed enough light on the organization of the consonant inventories, which unlike the vowel inventories, lack a complete and holistic explanation. The invariance of our measure implies that every inventory tries to be similar in terms of the measure, which leads us to argue that redundancy plays a very important role in shaping the structure of the consonant inventories. In order to validate this argument we determine the possibility of observing such an invariance if the consonant inventories had evolved by random chance. We find that the redundancy observed across the randomly generated inventories is substantially different from their real counterparts, which leads us to conclude that the invariance is not just "by-chance" and the measure that we define, indeed, largely governs the organizing principles of the consonant inventories.

[1] In phonology, features are the elements, which distinguish one phoneme from another. The features that distinguish the consonants can be broadly categorized into three different classes namely the *manner of articulation*, the *place of articulation* and *phonation*. Manner of articulation specifies how the flow of air takes place in the vocal tract during articulation of a consonant, whereas place of articulation specifies the active speech organ and also the place where it acts. Phonation describes the activity regarding the vibration of the vocal cords during the articulation of a consonant.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 104–111,
Prague, Czech Republic, June 2007.

Interestingly, this redundancy, when measured for the vowel inventories, does not exhibit any similar invariance. This immediately reveals that the principles that govern the formation of these two types of inventories are quite different in nature. Such an observation is significant since whether or not these principles are similar/different for the two inventories had been a question giving rise to perennial debate among the past researchers (Trubetzkoy, 1969/1939; Lindblom and Maddieson, 1988; Boersma, 1998; Clements, 2004). A possible reason for the observed dichotomy in the behavior of the vowel and consonant inventories with respect to redundancy can be as follows: while the organization of the vowel inventories is known to be governed by a single force - the *maximal perceptual contrast* (Jakobson, 1941; Liljencrants and Lindblom, 1972; de Boer, 2000)), consonant inventories are shaped by a complex interplay of several forces (Mukherjee et al., 2006). The invariance of redundancy, perhaps, reflects some sort of an equilibrium that arises from the interaction of these divergent forces.

The rest of the paper is structured as follows. In section 2 we briefly discuss the earlier works in connection to the sound inventories and then systematically build up the quantitative definition of redundancy from the linguistic theories that are already available in the literature. Section 3 details out the data source necessary for the experiments, describes the baseline for the experiments, reports the experiments performed, and presents the results obtained each time comparing the same with the baseline results. Finally we conclude in section 4 by summarizing our contributions, pointing out some of the implications of the current work and indicating the possible future directions.

2 Formulation of Redundancy

Linguistic research has documented a wide range of regularities across the sound systems of the world's languages. It has been postulated earlier by functional phonologists that such regularities are the consequences of certain general principles like *maximal perceptual contrast* (Liljencrants and Lindblom, 1972), which is desirable between the phonemes of a language for proper perception of each individual phoneme in a noisy environment, *ease of articulation* (Lindblom and Maddieson, 1988; de Boer, 2000), which requires that the sound systems of all languages are formed of certain universal (and highly frequent) sounds, and *ease of learnability* (de Boer, 2000), which is necessary for a speaker to learn the sounds of a language with minimum effort. In fact, the organization of the vowel inventories (especially those with a smaller size) across languages has been satisfactorily explained in terms of the single principle of maximal perceptual contrast (Jakobson, 1941; Liljencrants and Lindblom, 1972; de Boer, 2000).

On the other hand, in spite of several attempts (Lindblom and Maddieson, 1988; Boersma, 1998; Clements, 2004) the organization of the consonant inventories lacks a satisfactory explanation. However, one of the earliest observations about the consonant inventories has been that consonants tend to occur in pairs that exhibit strong correlation in terms of their features (Trubetzkoy, 1931). In order to explain these trends, *feature economy* was proposed as the organizing principle of the consonant inventories (Martinet, 1955). According to this principle, languages tend to maximize the combinatorial possibilities of a few distinctive features to generate a large number of consonants. Stated differently, a given consonant will have a higher than expected chance of occurrence in inventories in which all of its features have distinctively occurred in other consonants. The idea is illustrated, with an example, through Table 1. Various attempts have been made in the past to explain the aforementioned trends through linguistic insights (Boersma, 1998; Clements, 2004) mainly establishing their statistical significance. On the contrary, there has been very little work pertaining to the quantification of feature economy except in (Clements, 2004), where the author defines *economy index*, which is the ratio of the size of an inventory to the number of features that characterizes the inventory. However, this definition does not take into account the complexity that is involved in communicating the information about the inventory in terms of its constituent features.

Inspired by the aforementioned studies and the concepts of information theory (Shannon and Weaver, 1949) we try to quantitatively capture the amount of redundancy found across the consonant

plosive	voiced	voiceless
dental	/d/	/t/
bilabial	/b/	/p/

Table 1: The table shows four plosives. If a language has in its consonant inventory any three of the four phonemes listed in this table, then there is a higher than average chance that it will also have the fourth phoneme of the table in its inventory.

inventories in terms of their constituent features. Let us assume that we want to communicate the information about an inventory of size N over a transmission channel. Ideally, one should require $\log N$ bits to do the same (where the logarithm is with respect to base 2). However, since every natural system is to some extent redundant and languages are no exceptions, the number of bits actually used to encode the information is more than $\log N$. If we assume that the features are boolean in nature, then we can compute the number of bits used by a language to encode the information about its inventory by measuring the *entropy* as follows. For an inventory of size N let there be p_f consonants for which a particular feature f (where f is assumed to be boolean in nature) is present and q_f other consonants for which the same is absent. Thus the probability that a particular consonant chosen uniformly at random from this inventory has the feature f is $\frac{p_f}{N}$ and the probability that the consonant lacks the feature f is $\frac{q_f}{N}$ (=1–$\frac{p_f}{N}$). If F is the set of all features present in the consonants forming the inventory, then *feature entropy* F_E can be expressed as

$$F_E = \sum_{f \in F} (-\frac{p_f}{N} \log \frac{p_f}{N} - \frac{q_f}{N} \log \frac{q_f}{N}) \qquad (1)$$

F_E is therefore the measure of the minimum number of bits that is required to communicate the information about the entire inventory through the transmission channel. The lower the value of F_E the better it is in terms of the information transmission overhead. In order to capture the redundancy involved in the encoding we define the term *redundancy ratio* as follows,

$$RR = \frac{F_E}{\log N} \qquad (2)$$

which expresses the excess number of bits that is used by the constituent consonants of the inventory in terms of a ratio. The process of computing the value of RR for a hypothetical consonant inventory is illustrated in Figure 1.

I = {/b/, /d/, /g/} N = 3

F = {voiced, dental, bilabial, velar, plosive}

F	voiced	dental	bilabial	velar	plosive
/b/	1	0	1	0	1
/d/	1	1	0	0	1
/g/	1	0	0	1	1
p_f/N	1	0.33	0.33	0.33	1
q_f/N	0	0.67	0.67	0.67	0

F_E = 2.75

RR = F_E / log(N) = 1.74

Figure 1: The process of computing RR for a hypothetical inventory.

In the following section, we present the experimental setup and also report the experiments which we perform based on the above definition of redundancy. We subsequently show that redundancy ratio is invariant across the consonant inventories whereas the same is not true in the case of the vowel inventories.

3 Experiments and Results

In this section we discuss the data source necessary for the experiments, describe the baseline for the experiments, report the experiments performed, and present the results obtained each time comparing the same with the baseline results.

3.1 Data Source

Many typological studies (Ladefoged and Maddieson, 1996; Lindblom and Maddieson, 1988) of segmental inventories have been carried out in past on the UCLA Phonological Segment Inventory Database (UPSID) (Maddieson, 1984). UPSID gathers phonological systems of languages from all over the world, sampling more or less uniformly all the linguistic families. In this work we have used UPSID comprising of 317 languages and 541 consonants found across them, for our experiments.

3.2 Redundancy Ratio across the Consonant Inventories

In this section we measure the redundancy ratio (described earlier) of the consonant inventories of the languages recorded in UPSID. Figure 2 shows the scatter-plot of the redundancy ratio RR of each of the consonant inventories (y-axis) versus the inventory size (x-axis). The plot immediately reveals that the measure (i.e., RR) is almost invariant across the consonant inventories with respect to the inventory size. In fact, we can fit the scatter-plot with a straight line (by means of least square regression), which as depicted in Figure 2, has a negligible slope (m = – 0.018) and this in turn further confirms the above fact that RR is an invariant property of the consonant inventories with regard to their size. It is important to mention here that in this experiment we report the redundancy ratio of all the inventories of size less than or equal to 40. We neglect the inventories of the size greater than 40 since they are extremely rare (less than 0.5% of the languages of UPSID), and therefore, cannot provide us with statistically meaningful estimates. The same convention has been followed in all the subsequent experiments. Nevertheless, we have also computed the values of RR for larger inventories, whereby we have found that for an inventory size ≤ 60 the results are similar to those reported here. It is interesting to note that the largest of the consonant inventories Ga (size = 173) has an $RR = 1.9$, which is lower than all the other inventories.

The aforementioned claim that RR is an invariant across consonant inventories can be validated by performing a standard test of hypothesis. For this purpose, we randomly construct language inventories, as discussed later, and formulate a null hypothesis based on them.

Null Hypothesis: The invariance in the distribution of RRs observed across the real consonant inventories is also prevalent across the randomly generated inventories.

Having formulated the null hypothesis we now systematically attempt to reject the same with a very high probability. For this purpose we first construct random inventories and then perform a two sample t-test (Cohen, 1995) comparing the RRs of the real and the random inventories. The results show that

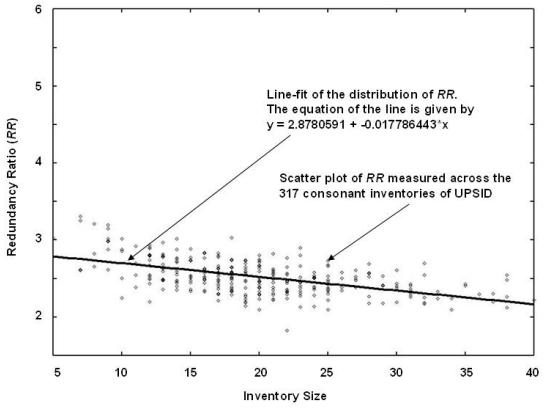


Figure 2: The scatter-plot of the redundancy ratio RR of each of the consonant inventories (y-axis) versus the inventory size (x-axis). The straight line-fit is also depicted by the bold line in the figure.

indeed the null hypothesis can be rejected with a very high probability. We proceed as follows.

3.2.1 Construction of Random Inventories

We employ two different models to generate the random inventories. In the first model the inventories are filled uniformly at random from the pool of 541 consonants. In the second model we assume that the distribution of the occurrence of the consonants over languages is known *a priori*. Note that in both of these cases, the size of the random inventories is same as its real counterpart. The results show that the distribution of RRs obtained from the second model has a closer match with the real inventories than that of the first model. This indicates that the occurrence frequency to some extent governs the law of organization of the consonant inventories. The detail of each of the models follow.

Model I – Purely Random Model: In this model we assume that the distribution of the consonant inventory size is known *a priori*. For each language inventory L let the size recorded in UPSID be denoted by s_L. Let there be 317 bins corresponding to each consonant inventory L. A bin corresponding to an inventory L is packed with s_L consonants chosen uniformly at random (without repetition) from the pool of 541 available consonants. Thus the consonant inventories of the 317 languages corresponding to the bins are generated. The method is summarized

in Algorithm 1.

for *I = 1 to 317* **do**
  for *size = 1 to* s_L **do**
    Choose a consonant c uniformly at random (without repetition) from the pool of 541 available consonants;
    Pack the consonant c in the bin corresponding to the inventory L;
  end
end

Algorithm 1: Algorithm to construct random inventories using Model I

Model II – Occurrence Frequency based Random Model: For each consonant c let the frequency of occurrence in UPSID be denoted by f_c. Let there be 317 bins each corresponding to a language in UPSID. f_c bins are then chosen uniformly at random and the consonant c is packed into these bins. Thus the consonant inventories of the 317 languages corresponding to the bins are generated. The entire idea is summarized in Algorithm 2.

for *each consonant c* **do**
  for *i = 1 to* f_c **do**
    Choose one of the 317 bins, corresponding to the languages in UPSID, uniformly at random;
    Pack the consonant c into the bin so chosen if it has not been already packed into this bin earlier;
  end
end

Algorithm 2: Algorithm to construct random inventories using Model II

3.2.2 Results Obtained from the Random Models

In this section we enumerate the results obtained by computing the RRs of the randomly generated inventories using Model I and Model II respectively. We compare the results with those of the real inventories and in each case show that the null hypothesis can be rejected with a significantly high probability.

Parameters	Real Inv.	Random Inv.
Mean	2.51177	3.59331
SDV	0.209531	0.475072

Parameters	Values
t	12.15
DF	66
p	$\leq$ 9.289e-17

Table 2: The results of the t-test comparing the distribution of RRs for the real and the random inventories (obtained through Model I). SDV: standard deviation, t: t-value of the test, DF: degrees of freedom, p: residual uncertainty.

Results from Model I: Figure 3 illustrates, for all the inventories obtained from 100 different simulation runs of Algorithm 1, the average redundancy ratio exhibited by the inventories of a particular size (y-axis), versus the inventory size (x-axis). The term "redundancy ratio exhibited by the inventories of a particular size" actually means the following. Let there be n consonant inventories of a particular inventory-size k. The average redundancy ratio of the inventories of size k is therefore given by $\frac{1}{n}\sum_{i=1}^{n} RR_i$ where RR_i signifies the redundancy ratio of the i^{th} inventory of size k. In Figure 3 we also present the same curve for the real consonant inventories appearing in UPSID. In these curves we further depict the error bars spanning the entire range of values starting from the minimum RR to the maximum RR for a given inventory size. The curves show that in case of real inventories the error bars span a very small range as compared to that of the randomly constructed ones. Moreover, the slopes of the curves are also significantly different. In order to test whether this difference is significant, we perform a t-test comparing the distribution of the values of RR that gives rise to such curves for the real and the random inventories. The results of the test are noted in Table 2. These statistics clearly shows that the distribution of RRs for the real and the random inventories are significantly different in nature. Stated differently, we can reject the null hypothesis with (100 - 9.29e-15)% confidence.

Results from Model II: Figure 4 illustrates, for all the inventories obtained from 100 different simu-

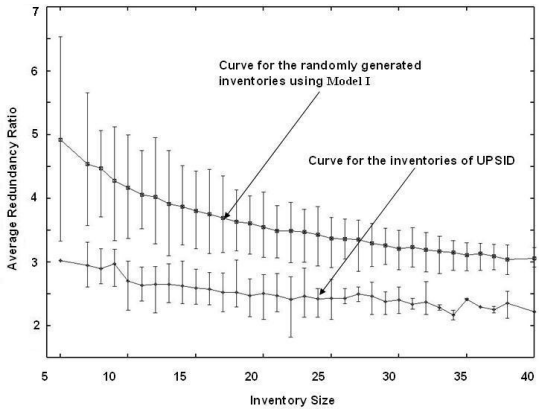


Figure 3: Curves showing the average redundancy ratio exhibited by the real as well as the random inventories (obtained through Model I) of a particular size (y-axis), versus the inventory size (x-axis).

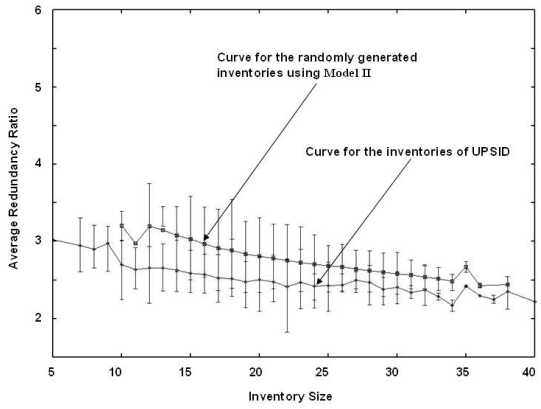


Figure 4: Curves showing the average redundancy ratio exhibited by the real as well as the random inventories (obtained through Model II) of a particular size (y-axis), versus the inventory size (x-axis).

lation runs of Algorithm 2, the average redundancy ratio exhibited by the inventories of a particular size (y-axis), versus the inventory size (x-axis). The figure shows the same curve for the real consonant inventories also. For each of the curve, the error bars span the entire range of values starting from the minimum RR to the maximum RR for a given inventory size. It is quite evident from the figure that the error bars for the curve representing the real inventories are smaller than those of the random ones. The nature of the two curves are also different though the difference is not as pronounced as in case of Model I. This is indicative of the fact that it is not only the occurrence frequency that governs the organization of the consonant inventories and there is a more complex phenomenon that results in such an invariant property. In fact, in this case also, the t-test statistics comparing the distribution of RRs for the real and the random inventories, reported in Table 3, allows us to reject the null hypothesis with (100–2.55e–3)% confidence.

Parameters	Real Inv.	Random Inv.
Mean	2.51177	2.76679
SDV	0.209531	0.228017

Parameters	Values
t	4.583
DF	60
p	$\leq$ 2.552e-05

Table 3: The results of the t-test comparing the distribution of RRs for the real and the random inventories (obtained through Model II).

3.3 Comparison with Vowel Inventories

Until now we have been looking into the organizational aspects of the consonant inventories. In this section we show that this organization is largely different from that of the vowel inventories in the sense that there is no such invariance observed across the vowel inventories unlike that of consonants. For this reason we start by computing the RRs of all the vowel inventories appearing in UPSID. Figure 5 shows the scatter plot of the redundancy ratio of each of the vowel inventories (y-axis) versus the inventory size (x-axis). The plot clearly indicates that the measure (i.e., RR) is not invariant across the vowel inventories and in fact, the straight line that fits the distribution has a slope of –0.14, which is around 10 times higher than that of the consonant inventories.

Figure 6 illustrates the average redundancy ratio exhibited by the vowel and the consonant inventories of a particular size (y-axis), versus the inventory size (x-axis). The error bars indicating the variability of RR among the inventories of a fixed size also span a much larger range for the vowel inventories than for the consonant inventories.

The significance of the difference in the nature of the distribution of RRs for the vowel and the consonant inventories can be again estimated by performing a t-test. The null hypothesis in this case is as follows.

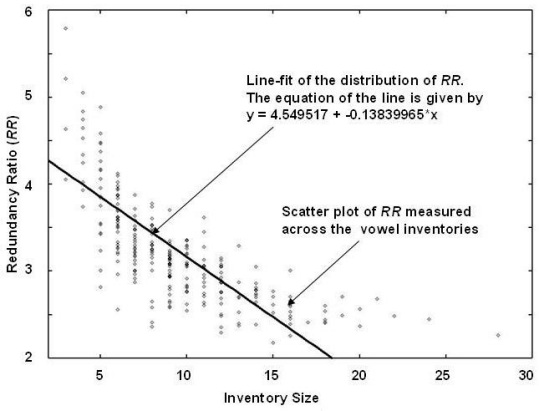


Figure 5: The scatter-plot of the redundancy ratio RR of each of the vowel inventories (y-axis) versus the inventory size (x-axis). The straight line-fit is depicted by the bold line in the figure.

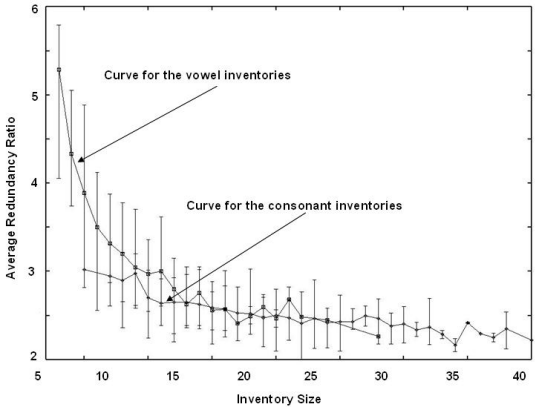


Figure 6: Curves showing the average redundancy ratio exhibited by the vowel as well as the consonant inventories of a particular size (y-axis), versus the inventory size (x-axis).

Null Hypothesis: The nature of the distribution of RRs for the vowel and the consonant inventories is same.

We can now perform the t-test to verify whether we can reject the above hypothesis. Table 4 presents the results of the test. The statistics immediately confirms that the null hypothesis can be rejected with 99.932% confidence.

Parameters	Consonant Inv.	Vowel Inv.
Mean	2.51177	2.98797
SDV	0.209531	0.726547

Parameters	Values
t	3.612
DF	54
p	≤ 0.000683

Table 4: The results of the t-test comparing the distribution of RRs for the consonant and the vowel inventories.

4 Conclusions, Discussion and Future Work

In this paper we have mathematically captured the redundancy observed across the sound inventories of the world's languages. We started by systematically defining the term redundancy ratio and measuring the value of the same for the inventories. Some of our important findings are,

1. Redundancy ratio is an invariant property of the consonant inventories with respect to the inventory size.

2. A more complex phenomenon than merely the occurrence frequency results in such an invariance.

3. Unlike the consonant inventories, the vowel inventories are not indicative of such an invariance.

Until now we have concentrated on establishing the invariance of the redundancy ratio across the consonant inventories rather than reasoning why it could have emerged. One possible way to answer this question is to look for the error correcting capability of the encoding scheme that nature had employed for characterization of the consonants. Ideally, if redundancy has to be invariant, then this capability should be almost constant. As a proof of concept we randomly select a consonant from inventories of different size and compute its hamming distance from the rest of the consonants in the inventory. Figure 7 shows for a randomly chosen consonant c from an inventory of size 10, 15, 20 and 30 respectively, the number of the consonants at a particular hamming distance from c (y-axis) versus the hamming distance (x-axis). The curve clearly indicates that majority of the consonants are at a hamming distance of 4 from c, which in turn implies that the encoding scheme has almost a fixed error correcting capability of 1 bit. This can be the precise reason behind the invariance of the redundancy ra-

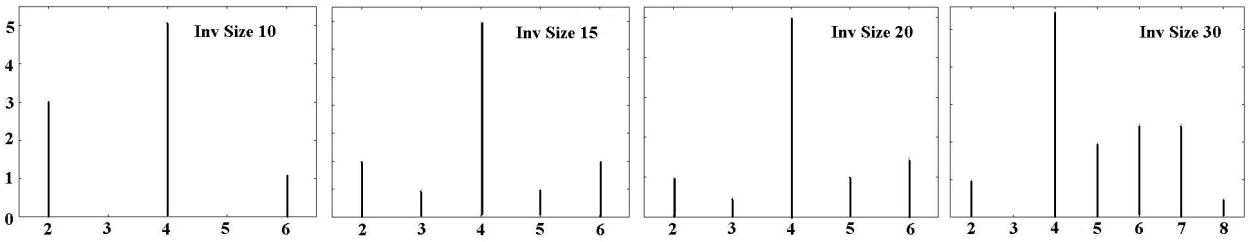


Figure 7: Histograms showing the the number of consonants at a particular hamming distance (y-axis), from a randomly chosen consonant c, versus the hamming distance (x-axis).

tio. Initial studies into the vowel inventories show that for a randomly chosen vowel, its hamming distance from the other vowels in the same inventory varies with the inventory size. In other words, the error correcting capability of a vowel inventory seems to be dependent on the size of the inventory.

We believe that these results are significant as well as insightful. Nevertheless, one should be aware of the fact that the formulation of RR heavily banks on the set of features that are used to represent the phonemes. Unfortunately, there is no consensus on the set of representative features, even though there are numerous suggestions available in the literature. However, the basic concept of RR and the process of analysis presented here is independent of the choice of the feature set. In the current study we have used the binary features provided in UPSID, which could be very well replaced by other representations, including multi-valued feature systems; we look forward to do the same as a part of our future work.

References

B. de Boer. 2000. Self-organisation in vowel systems. *Journal of Phonetics*, 28(4), 441–465.

P. Boersma. 1998. *Functional phonology*, Doctoral thesis, University of Amsterdam, The Hague: Holland Academic Graphics.

N. Clements. 2004. Features and sound inventories. *Symposium on Phonological Theory: Representations and Architecture*, CUNY.

P. R. Cohen. 1995. *Empirical methods for artificial intelligence*, MIT Press, Cambridge.

L. L. Gatlin. 1974. Conservation of Shannon's redundancy for proteins *Jour. Mol. Evol.*, 3, 189–208.

R. Jakobson. 1941. *Kindersprache, aphasie und allgemeine lautgesetze*, Uppsala, Reprinted in *Selected Writings I. Mouton*, The Hague, 1962, 328-401.

D. C. Krakauer and J. B. Plotkin. 2002. Redundancy, antiredundancy, and the robustness of genomes. *PNAS*, 99(3), 1405-1409.

A. M. Lesk. 2002. *Introduction to bioinformatics*, Oxford University Press, New York.

P. Ladefoged and I. Maddieson. 1996. *Sounds of the world's languages*, Oxford: Blackwell.

J. Liljencrants and B. Lindblom. 1972. Numerical simulation of vowel quality systems: the role of perceptual contrast. *Language*, 48, 839–862.

B. Lindblom and I. Maddieson. 1988. Phonetic universals in consonant systems. *Language, Speech, and Mind*, 62–78.

I. Maddieson. 1984. *Patterns of sounds*, Cambridge University Press, Cambridge.

A. Martinet 1955. *Èconomie des changements phonétiques*, Berne: A. Francke.

A. Mukherjee, M. Choudhury, A. Basu and N. Ganguly. 2006. Modeling the co-occurrence principles of the consonant inventories: A complex network approach. *arXiv:physics/0606132 (preprint)*.

C. E. Shannon and W. Weaver. 1949. *The mathematical theory of information*, Urbana: University of Illinois Press.

N. Trubetzkoy. 1931. Die phonologischen systeme. *TCLP*, 4, 96–116.

N. Trubetzkoy. 1969. *Principles of phonology*, Berkeley: University of California Press.

A. Woollard. 2005. Gene duplications and genetic redundancy in C. *elegans*, *WormBook*.

Multilingual Transliteration Using Feature based Phonetic Method

Su-Youn Yoon, Kyoung-Young Kim and Richard Sproat
University of Illinois at Urbana-Champaign
{syoon9,kkim36,rws}@uiuc.edu

Abstract

In this paper we investigate named entity transliteration based on a phonetic scoring method. The phonetic method is computed using phonetic features and carefully designed pseudo features. The proposed method is tested with four languages – Arabic, Chinese, Hindi and Korean – and one source language – English, using comparable corpora. The proposed method is developed from the phonetic method originally proposed in Tao et al. (2006). In contrast to the phonetic method in Tao et al. (2006) constructed on the basis of pure linguistic knowledge, the method in this study is trained using the Winnow machine learning algorithm. There is salient improvement in Hindi and Arabic compared to the previous study. Moreover, we demonstrate that the method can also achieve comparable results, when it is trained on language data different from the target language. The method can be applied both with minimal data, and without target language data for various languages.

1 Introduction.

In this paper, we develop a multi-lingual transliteration system for named entities. Named entity transliteration is the process of producing, for a name in a source language, a set of one or more transliteration candidates in a target language. The correct transliteration of named entities is crucial, since they are frequent and important key words in information retrieval. In addition, requests in retrieving relevant documents in multiple languages require the development of the multi-lingual system.

The system is constructed using paired comparable texts. The comparable texts are about the same or related topics, but are not, in general, translations of each other. Using this data, the transliteration method aims to find transliteration correspondences in the paired languages. For example, if there were an English and Arabic newspaper on the same day, each of the newspapers would contain articles about the same important international events. From these comparable articles across the paired languages, the same named entities are expected to be found. Thus, from the named entities in an English newspaper, the method would find transliteration correspondences in comparable texts in other languages.

The multi-lingual transliteration system entails solving several problems which are very challenging. First, it should show stable performance for many unrelated languages. The transliteration will be influenced by the difference in the phonological systems of the language pairs, and the process of transliteration differs according to the languages involved. For example, in Arabic texts, short vowels are rarely written while long vowels are written. When transliterating English names, the vowels are disappeared or written as long vowels. For example *London* is transliterated as *lndn* لنـــدن, and both vowels are not represented in the transliteration. However, *Washington* is often transliterated as *wSnjTwn* واشـــــــنطون , and the final vowel is realized with long vowel. Transliterations in Chinese are very different from the original English pronunciation due to the

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 112–119,
Prague, Czech Republic, June 2007.

limited syllable structure and phoneme inventory of Chinese. For example, Chinese does not allow consonant clusters or coda consonants except [n,ŋ], and this results in deletion, substitution of consonants or insertion of vowels. Thus while a syllable initial /d/ may surface as in *Baghdad* 巴格达 *ba-ge-da*, note that the syllable final /d/ is not represented. Multi-lingual transliteration system should solve these language dependent characteristics.

One of the most important concerns in a multilingual transliteration system is its applicability given a small amount of training data, or even no training data: for arbitrary language pairs, one cannot in general assume resources such as name dictionaries. Indeed, for some rarely spoken languages, it is practically impossible to find enough training data. Therefore, the proposed method aims to obtain comparable performance with little training data.

2 Previous Work

Previous work — e.g. (Knight and Graehl, 1998; Meng et al., 2001; Al-Onaizan and Knight, 2002; Gao et al., 2004) — has mostly assumed that one has a training lexicon of transliteration pairs, from which one can learn a model, often a source-channel or MaxEnt-based model.

Comparable corpora have been studied extensively in the literature, but transliteration in the context of comparable corpora has not been well addressed. In our work, we adopt the method proposed in (Tao et al., 2006) and apply it to the problem of transliteration.

Measuring phonetic similarity between words has been studied for a long time. In many studies, two strings are aligned using a string alignment algorithm, and an edit distance (the sum of the cost for each edit operation), is used as the phonetic distance between them. The resulting distance depends on the costs of the edit operation. There are several approaches that use distinctive features to determine the costs of the edit operation. Gildea and Jurafsky (1996) counted the number of features whose values are different, and used them as a substitution cost. However, this approach has a crucial limitation: the cost does not consider the importance of the features. Nerbonne and Heeringa (1997) assigned a weight for each feature based on entropy and information gain, but the results were even less accurate than the method without weight.

3 Phonetic transliteration method

In this paper, the phonetic transliteration is performed using the following steps:

1) **Generation of the pronunciation for English words and target words**:

a. Pronunciations for English words are obtained using the Festival text-to-speech system (Taylor et al., 1998).

b. Target words are automatically converted into their phonemic level transcriptions by various language-dependent means. In the case of Mandarin Chinese, this is based on the standard Pinyin transliteration system. Arabic words are converted based on orthography, and the resulting transcriptions are reasonably correct except for the fact that short vowels were not represented. Similarly, the pronunciation of Hindi and Korean can be well-approximated based on the standard orthographic representation. All pronunciations are based on the WorldBet transliteration system (Hieronymus, 1995), an ascii-only version of the IPA.

2) **Training a linear classifier using the Winnow algorithm**:

A linear classifier is trained using the training data which is composed of transliteration pairs and non-transliteration pairs. Transliteration pairs are extracted from the transliteration dictionary, while non-transliteration pairs are composed of an English named entity and a random word from the target language newspaper.

a. For all the training data, the pairs of pronunciations are aligned using standard string alignment algorithm based on Kruskal (1999). The substitution/insertion/deletion cost for the string alignment algorithm is based on the baseline cost from (Tao et al, 2006).

b. All phonemes in the pronunciations are decomposed into their features. The features used in this study will be explained in detail in part 3.1.

c. For every phoneme pair (p_1, p_2) in the aligned pronunciations, a feature x_i has a '+1' value or a '–1' value:

$$x_i = \begin{cases} +1 & \text{when } p_1 \text{ and } p_2 \text{ have the same values for feature } x_i \\ -1 & \text{otherwise} \end{cases}$$

d. A linear classifier is trained using the Winnow algorithm from the SNoW toolkit (Carlson et al., 1999).

3) **Scoring English-target word pair:**

a. For a given English word, the score between it and a target word is computed using the linear classifier.

b. The score ranges from 0 to any positive number, and the candidate with the highest score is selected as the transliteration of the given English name.

3.1 Feature set

Halle and Clements (1983)'s distinctive features are used in order to model the substitution/ insertion/deletion costs for the string-alignment algorithm and linear classifier. A distinctive feature is a feature that describes the phonetic characteristics of phonetic segments.

However, distinctive features alone are not enough to model the frequent sound change patterns that occur when words are adapted across languages. For example, stop and fricative consonants such as /p, t, k, b, d, g, s, z/ are frequently deleted when they appear in the coda position. This tendency is extremely salient when the target languages do not allow coda consonants or consonant clusters. For example, since Chinese only allows /n, ŋ/ in coda position, stop consonants in the coda position are frequently lost; *Stanford* is transliterated as *sitanfu*, with the final /d/ lost. Since traditional distinctive features do not consider the position in the syllable, this pattern cannot be captured by distinctive features alone. To capture these sound change patterns, additional features such as "deletion of stop/fricative consonant in the coda position" must be considered.

Based on the pronunciation error data of learners of English as a second language as reported in (Swan and Smith, 2002), we propose the use of what we will term *pseudofeatures.* The pseudo features in this study are same as in Tao et al. (2006). Swan & Smith (2002)'s study covers 25 languages including Asian languages such as Thai, Korean, Chinese and Japanese, European languages such as German, Italian, French and Polish, and Middle East languages such as Arabic and Farsi. The substitution/insertion/deletion errors of phonemes were collected from this data. The following types of errors frequently occur in second language learners' speech production.

(1) **Substitution**: If the learner's first language does not have a particular phoneme found in English, it is substituted by the most similar phoneme in their first language.

(2) **Insertion**: If the learner's first language does not have a particular consonant cluster in English, a vowel is inserted.

(3) **Deletion**: If the learner's first language does not have a particular consonant cluster in English, one consonant in the consonant cluster is deleted.

The same substitution/deletion/insertion patterns in a second language learner's errors also appear in the transliteration of foreign names. The deletion of the stop consonant which appears in English-Chinese transliterations occurs frequently in the English pronunciation spoken by Chinese speakers. Therefore, the error patterns in second language learners' can be used in transliteration.

Based on (1) ~ (3), 21 pseudo features were designed. All features have binary values. Using these 21 pseudo features and 20 distinctive features, a linear classifier is trained. Some examples of pseudo features are presented in Table 1.

Pseudo-Feature	Description	Example
Consonant-coda	Substitution of consonant feature in coda position	
Sonorant-coda	Substitution of sonorant feature in coda position	Substitution between [ŋ] and [g] in coda position in Arabic
Labial-coda	Substitution of labial feature in coda position	Substitution between [m] and [n] in coda position in Chinese
j-exception	Substitution of [j] and [dʒ]	Spanish/Catalan and Festival error
w-exception	Substitution of [v] and [w]	Chinese/Farsi and Festival error

Table 1. Examples of pseudo features

3.2 Scoring the English-target word pair

A linear classifier is trained using the Winnow algorithm from the SNoW toolkit.

The Winnow algorithm is one of the update rules for linear classifier. A linear classifier is an algorithm to find a linear function that best separates the data. For the set of features X and set of weights W, the linear classifier is defined as [1] (Mitchell, T., 1997)

$$X = \{x_1, x_2, \ldots x_n\}$$
$$W = \{w_1, w_2, \ldots w_n\} \qquad [1]$$
$$f(x) = \begin{cases} 1 \text{ if } w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n > 0 \\ -1 \text{ otherwise} \end{cases}$$

The linear function assigns label +1 when the paired target language word is the transliteration of given English word, while it assigns label –1 when it is not a transliteration of given English word.

The score of an English word and target word pair is computed using equation [2] which is part of the definition of *f(x)* in equation [1].

$$w_0 + \sum_{i=1}^{n} w_i x_i \qquad [2]$$

The output of equation [2] is termed *the target node activation*. If this value is high, class 1 is more activated, and the pair is more likely to be a transliteration pair. To illustrate, let us assume there are two candidates in target language (t_1 and t_2) for an English word e. If the score of (e, t_1) is higher than the score of (e, t_2), the pair (e, t_1) has stronger activation than (e, t_2). It means that t_1 scores higher as the transliteration of e than t_2. Therefore, the candidate with the highest score (in this case t_1) is selected as the transliteration of the given English name.

4 Experiment and Results

The linear function was trained for each language, separately. 500 transliteration pairs were randomly selected from each transliteration dictionary, and used as positive examples in the training procedure. This is quite small compared to previous approaches such as Knight and Graehl (1998) or Gao et al. (2004). In addition, 1500 words were randomly selected from the newspaper in the target languages, and paired with English words in the positive examples. A total of 750,000 pairs (500 English words × 1500 target words) were generated, and used as negative examples in the training procedure.

Table 2 presents the source of training data for each language.

	Transliteration pair	Target word
Arabic	New Mexico State University	Xinhua Arabic newswire
Chinese	Behavior Design Corporation	Xinhua Chinese newswire
Hindi	Naidunia Hindi newswire	Naidunia Hindi newswire
Korean	the National Institute of the Korean language	Chosun Korean newspaper

Table 2. Sources of the training data

The phonetic transliteration method was evaluated using comparable corpora, consisting of newspaper articles in English and the target languages—Arabic, Chinese, Hindi, and Korean– from the same day, or almost the same day. Using comparable corpora, the named-entities for persons and locations were extracted from the English text; in this paper, the English named-entities were extracted using the named-entity recognizer described in Li et al. (2004), based on the SNoW machine learning toolkit (Carlson et al., 1999).

The transliteration task was performed using the following steps:

1) English text was tagged using the named-entity recognizer. The 200 most frequent named entities were extracted from seven days' worth of the English newswire text. Among pronunciations of words generated by the Festival text-to speech system, 3% contained errors representing monophthongs instead of diphthongs or vice versa. 1.5% of all cases misrepresented single consonant, and 6% showed errors in the vowels. Overall, 10.5% of the tokens contained pronunciation errors which could trigger errors in transliteration.

2) To generate the Arabic and Hindi candidates, all words from the same seven days were extracted. In the case of Korean corpus, the collection of newspapers was from every five days, unlike the other three language corpora which were collected every day; therefore, candidates of Korean were

generated from one month of newspapers, since seven days of newspaper articles did not show a sufficient number of transliteration candidates. This caused the total number of candidates to be much bigger than for the other languages.

The words were stemmed all possible ways using simple hand-developed affix lists: for example, given a Hindi word c1c2c3, if both c3 and c2c3 are in the suffix and ending list, then this single word generated three possible candidates: c1, c1c2, and c1c2c3.

3) Segmenting Chinese sentences requires a dictionary or supervised segmenter. Since the goal is to use minimal knowledge or data from the target language, using supervised methods is inappropriate for our approach. Therefore, Chinese sentences were not segmented. Using the 495 characters that are frequently used for transliterating foreign names (Sproat et al., 1996), a sequence of three of more characters from the list was taken as a possible candidate for Chinese.

4) For the given 200 English named entities and target language candidate lists, all the possible pairings of English and target-language name were considered as possible transliteration pairs.

The number of candidates for each target language is presented in Table 3.

Language	The number of candidates
Arabic	12,466
Chinese	6,291
Hindi	10,169
Korean	42,757

Table 3. Number of candidates for each target language.

5) Node activation scores were calculated for each pair in the test data, and the candidates were ranked by their score. The candidate with the highest node activation score was selected as the transliteration of the given English name.

Some examples of English words and the top three ranking candidates among all of the potential target-language candidates were given in Tables 4, 5. Starred entries are correct.

English Word	Rank	Candidate	
		Script	Romanization
Arafat	*1 2 3	阿拉法特 拉法地奥 拉维奇	a-la-fa-te la-fa-di-ao la-wei-qi

Table 4. Examples of the top-3 candidates in the transliteration of English – Chinese

English Word	Rank	Candidate	
		Script	Romanization
Vietnam	*1	베트남	be-thu-nam
	2	베트남측	be-thu-nam-chug
	3	표준어와	pyo-jun-e-wa
Australia	*1	오스트레일리아	o-su-thu-ley-il-li-a
	2	웃돌아	us-tol-la
	3	오스트레일리아에서	o-su-thu-ley-il-li-a-ey-se

Table 5. Examples of the top-3 candidates in the transliteration of English-Korean

To evaluate the proposed transliteration methods quantitatively, the Mean Reciprocal Rank (MRR), a measure commonly used in information retrieval when there is precisely one correct answer (Kandor and Vorhees, 2000) was measured, following Tao and Zhai (2005).

Since the evaluation data obtained from the comparable corpus was small, the systems were evaluated using both held-out data from the transliteration dictionary and comparable corpus.

First, the results of the held-out data will be presented. For a given English name and target language candidates, all possible combinations were generated. Table 6 presents the size of held-out data, and Table 7 presents MRR of the held-out data.

	Number of English named entities	Number of Candidates in target language	Number of total pairs used in the evaluation
Arabic	500	1,500	750,000
Chinese	500	1,500	750,000
Hindi	100	1,500	150,000
Korean	100	1,500	150,000

Table 6. Size of the test data

	Baseline	Winnow	
		Total feature	distinctive feature only
Arabic	0.66	0.74	0.70
Chinese	0.74	0.74	0.72
Hindi	0.87	0.91	0.91
Korean	0.82	0.85	0.82

Table 7. MRRs of the phonetic transliteration

The baseline was computed using the phonetic transliteration method proposed in Tao et al. (2006). In contrast to the method in this study, the baseline system is purely based on linguistic knowledge. In the baseline system, the edit distance, which was the result of the string alignment algorithm, was used as the score of an English-target word pair. The performance of the edit distance was dependent on insertion/deletion/ substitution costs. These costs were determined based on the distinctive features and pseudo features, based on the pure linguistic knowledge without training data. As illustrated in Table 7, the phonetic transliteration method using features worked adequately for multilingual data, as phonetic features are universal, unlike the phonemes which are composed of them. Adopting phonetic features as the units for transliteration yielded the baseline performance.

In order to evaluate the effectiveness of pseudo features, the method was trained using two different feature sets: a total feature set and a distinctive feature-only set. For Arabic, Chinese and Korean, the MRR of the total feature set was higher than the MRR of the distinctive feature-only set. The improvement of the total set was 4% for Arabic, 2.6% for Chinese, 2.4% for Korean. There was no improvement of the total set in Hindi. In general, the pseudo features improved the accuracy of the transliteration.

For all languages, the MRR of the Winnow algorithm with the total feature set was higher than the baseline. There was 7% improvement for Arabic, 0.7% improvement for Chinese, 4% improvement for Hindi and 3% improvement for Korean.

We turn now to the results on comparable corpora. We attempted to create a complete set of answers for the 200 English names in our test set, but part of the English names did not seem to have any standard transliteration in the target language according to the native speaker's judgment. Accordingly, we removed these names from the evaluation set. Thus, the resulting list was less than 200 English names, as shown in the second column of Table 8; (Table 8 *All*). Furthermore, some correct transliterations were not found in our candidate list for the target languages, since the answer never occurred in the target news articles; (Table 8 *Missing*). Thus this results in a smaller number of candidates to evaluate. This smaller number is given in the fourth column of Table 8; (Table 8 *Core*).

Language	# All	# Missing	#Core
Arabic	192	121	71
Chinese	186	92	94
Hindi	144	83	61
Korean	195	114	81

Table 8. Number of evaluated English Name

MRRs were computed on the two sets represented by the count in column 2, and the smaller set represented by the count in column 4. We termed the former MRR "AllMRR" and the latter "CoreMRR". In Table 9, "CoreMRR" and "AllMRR" of the method were presented.

	Baseline		Winnow	
	All-MRR	Core MRR	All-MRR	Core MRR
Arabic	0.20	0.53	0.22	0.61
Chinese	0.25	0.49	0.25	0.50
Hindi	0.30	0.69	0.36	0.86
Korean	0.30	0.71	0.29	0.69

Table 9. MRRs of the phonetic transliteration

		test data			
		Arabic	Chinese	Hindi	Korean
training data	Arabic	**0.61**	0.50	0.86	0.63
	Chinese	0.59	**0.50**	0.80	0.66
	Hindi	0.59	0.54	**0.86**	0.67
	Korean	0.56	0.51	0.76	**0.69**

Table 10. MRRs for the phonetic transliteration 2

In both methods, CoreMRRs were higher than 0.49 for all languages. That is, if the answer is in the target language texts, then the method finds the correct answer within the top 2 words.

As with the previously discussed results, there were salient improvements in Arabic and Hindi when using the Winnow algorithm. The MRRs of the Winnow algorithm except Korean were higher than the baseline. There was 7% improvement for Arabic and 17% improvement for Hindi in CoreMRR. In contrast to the 3% improvement in held-out data, there was a 2% decrease in Korean: the MRRs of Korean from the Winnow algorithm were lower than baseline, possibly because of the limited size of the evaluation data. Similar to the results of held-out data, the improvement in Chinese was small (1%).

The MRRs of Hindi and the MRRs of Korean were higher than the MRRs of Arabic and Chinese. The lower MRRs of Arabic and Chinese may result from the phonological structures of the languages. In general, transliteration of English word into Arabic and Chinese is much more irregular than the transliteration into Hindi and Korean in terms of phonetics.

To test the applicability to languages for which training data is not available, we also investigated the use of models trained on language pairs *different from* the target language pair. Thus, for each test language pair, we evaluated the performance of models trained on each of the other language pairs. For example, three models were trained using Chinese, Hindi, and Korean, and they were tested with Arabic data. The CoreMRRs of this experiment were presented in Table 10. Note that the diagonal in this Table represents the within-language-pair training and testing scenario that we reported on above.

For Arabic, Hindi, and Korean, MRRs were indeed the highest when the methods were trained using data from the same language, as indicated by the boldface MRR scores on the diagonal. In general, however, the MRRs were not saliently lower across the board when using different language data than using same-language data in training and testing. For all languages, MRRs for the cross-language case were best when the methods were trained using Hindi. The differences between MRRs of the method trained from Hindi and MRRs of the method by homogeneous language data were 2% for Arabic and Korean. In the case of Chinese, MRRs of the method trained by Hindi was actually *better* than MRRs obtained by Chinese training data. Hindi has a large phoneme inventory compared to Korean, Arabic, and Chinese, so the relationship between English phonemes and Hindi phonemes is relatively regular, and only small number of language specific transliteration rules exist. That is, the language specific influences from Hindi are smaller than those from other languages. This characteristic of Hindi may result in the high MRRs for other languages. What these results imply is that named entity transliteration could be performed without training data for the target language with phonetic feature as a unit. This approach is especially valuable for languages for which training data is minimal or lacking.

5 Conclusion

In this paper, a phonetic method for multilingual transliteration was proposed. The method was based on string alignment, and linear classifiers trained using the Winnow algorithm. In order to learn both language-universal and language-specific transliteration characteristics, distinctive

features and pseudo features were used in training. The method can be trained using a small amount of training data, and the performance decreases only by a small degree when it is trained with a language different from the test data. Therefore, this method is extremely useful for underrepresented languages for which training data is difficult to find.

Acknowledgments

This work was funded the National Security Agency contract NBCHC040176 (REFLEX) and a Google Research grant.

References

Y. Al-Onaizan and K. Knight. 2002. Machine transliteration of names in Arabic text. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, Philadelphia, PA.

Andrew J. Carlson, Chad M. Cumby, Jeff L. Rosen, and Dan Roth. 1999. The SNoW learning architecture. *Technical Report UIUCDCS-R-99-2101*, UIUC CS Dept.

Wei Gao, Kam-Fai Wong, and Wai Lam. 2004. Phoneme based transliteration of foreign names for OOV problem. *Proceeding of IJCNLP*, 374–381.

Daniel Gildea and Daniel Jurafsky. 1996. Learning Bias and Phonological-Rule Induction. *Computational Linguistics* 22(4):497–530.

Morris Halle and G.N. Clements. 1983. *Problem book in phonology*. MIT press, Cambridge.

James Hieronymus. 1995. *Ascii phonetic symbols for the world's languages: Worldbet.* http://www.ling.ohio-tate.edu/ edwards/worldbet.pdf.

Paul B. Kantor and Ellen B. Voorhees. 2000. The TREC-5 confusion track: Comparing retrieval methods for scanned text. *Information Retrieval*, 2: 165–176.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4).

Joseph B. Kruskal. 1999. An overview of sequence comparison. *Time Warps, String Edits, and Macromolecules*, CSLI, 2nd edition, 1–44.

Xin Li, Paul Morie, and Dan Roth. 2004. Robust reading: Identification and tracing of ambiguous names. *Proceeding of NAACL-2004*.

H.M. Meng, W.K Lo, B. Chen, and K. Tang. 2001. Generating phonetic cognates to handle named entities in English-Chinese cross-language spoken document retrieval. In *Proceedings of the Automatic Speech Recognition and Understanding Workshop*.

Tom M. Mitchell. 1997. *Machine Learnin*g, McCraw-Hill, Boston.

John Nerbonne and Wilbert Heeringa. 1997. Measuring Dialect Distance Phonetically. *Proceedings of the 3rd Meeting of the ACL Special Interest Group in Computational Phonology*.

Richard Sproat, Chilin. Shih, William A. Gale, and Nancy Chang. 1996. A stochastic finite-state word-segmentation algorithm for Chinese. *Computational Linguistics*, 22(3).

Michael Swan and Bernard Smith. 2002. *Learner English*, Cambridge University Press, Cambridge .

Tao Tao and ChengXiang Zhai. 2005. Mining comparable bilingual text corpora for cross-language information integration. *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 691–696.

Tao Tao, Su-Youn Yoon, Andrew Fister, Richard Sproat and ChengXiang Zhai. "Unsupervised Named Entity Transliteration Using Temporal and Phonetic Correlation." EMNLP, July 22-23, 2006, Sydney, Australia.

Paul A. Taylor, Alan Black, and Richard Caley. 1998. The architecture of the Festival speech synthesis system. *Proceedings of the Third ESCAWorkshop on SpeechSynthesis*, 147–151.

Semantic Transliteration of Personal Names

Haizhou Li*, Khe Chai Sim*, Jin-Shea Kuo†, Minghui Dong*

*Institute for Infocomm Research
Singapore 119613
{hli,kcsim,mhdong}@i2r.a-star.edu.sg

†Chung-Hwa Telecom Laboratories
Taiwan
jskuo@cht.com.tw

Abstract

Words of foreign origin are referred to as borrowed words or loanwords. A loanword is usually imported to Chinese by phonetic transliteration if a translation is not easily available. Semantic transliteration is seen as a good tradition in introducing foreign words to Chinese. Not only does it preserve how a word sounds in the source language, it also carries forward the word's original semantic attributes. This paper attempts to automate the semantic transliteration process for the first time. We conduct an inquiry into the feasibility of semantic transliteration and propose a probabilistic model for transliterating personal names in Latin script into Chinese. The results show that semantic transliteration substantially and consistently improves accuracy over phonetic transliteration in all the experiments.

1 Introduction

The study of Chinese transliteration dates back to the seventh century when Buddhist scriptures were translated into Chinese. The earliest bit of Chinese translation theory related to transliteration may be the principle of "Names should follow their bearers, while things should follow Chinese." In other words, names should be transliterated, while things should be translated according to their meanings. The same theory still holds today.

Transliteration has been practiced in several ways, including phonetic transliteration and phonetic-semantic transliteration. By phonetic transliteration, we mean rewriting a foreign word in native grapheme such that its original pronunciation is preserved. For example, *London* becomes 伦敦 /Lun-Dun/[1] which does not carry any clear connotations. Phonetic transliteration represents the common practice in transliteration. Phonetic-semantic transliteration, hereafter referred to as semantic transliteration for short, is an advanced translation technique that is considered as a recommended translation practice for centuries. It translates a foreign word by preserving both its original pronunciation and meaning. For example, Xu Guangqi[2] translated *geo-* in *geometry* into Chinese as 几何 /Ji-He/, which carries the pronunciation of *geo-* and expresses the meaning of "a science concerned with measuring the earth".

Many of the loanwords exist in today's Chinese through semantic transliteration, which has been well received (Hu and Xu, 2003; Hu, 2004) by the people because of many advantages. Here we just name a few. (1) It brings in not only the sound, but also the meaning that fills in the semantic blank left by phonetic transliteration. This also reminds people that it is a loanword and avoids misleading; (2) It provides etymological clues that make it easy to trace back to the root of the words. For example, a transliterated Japanese name will maintain its Japanese identity in its Chinese appearance; (3) It evokes desirable associations, for example, an English girl's name is transliterated with Chinese characters that have clear feminine association, thus maintaining the gender identity.

[1] Hereafter, Chinese characters are also denoted in *Pinyin* romanization system, for ease of reference.

[2] Xu Quangqi (1562–1633) translated The Original Manuscript of Geometry to Chinese jointly with Matteo Ricci.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 120–127,
Prague, Czech Republic, June 2007.

Unfortunately, most of the reported work in the area of machine transliteration has not ventured into semantic transliteration yet. The Latin-scripted personal names are always assumed to homogeneously follow the English phonic rules in automatic transliteration (Li et al., 2004). Therefore, the same transliteration model is applied to all the names indiscriminatively. This assumption degrades the performance of transliteration because each language has its own phonic rule and the Chinese characters to be adopted depend on the following semantic attributes of a foreign name.

(1) Language of origin: An English word is not necessarily of pure English origin. In English news reports about Asian happenings, an English personal name may have been originated from Chinese, Japanese or Korean. The language origin affects the phonic rules and the characters to be used in transliteration[3]. For example, a Japanese name *Matsumoto* should be transliterated as 松本 /Song-Ben/, instead of 马茨莫托 /Ma-Ci-Mo-Tuo/ as if it were an English name.

(2) Gender association: A given name typically implies a clear gender association in both the source and target languages. For example, the Chinese transliterations of *Alice* and *Alexandra* are 爱丽丝 /Ai-Li-Si/ and 亚历山大 /Ya-Li-Shan-Da/ respectively, showing clear feminine and masculine characteristics. Transliterating *Alice* as 埃里斯 /Ai-Li-Si/ is phonetically correct, but semantically inadequate due to an improper gender association.

(3) Surname and given name: The Chinese name system is the original pattern of names in Eastern Asia such as China, Korea and Vietnam, in which a limited number of characters[4] are used for surnames while those for given names are less restrictive. Even for English names, the character set for given name transliterations are different from that for surnames.

Here are two examples of semantic transliteration for personal names. *George Bush* and *Yamamoto Akiko* are transliterated into 乔治␣布什 and 山本 亚喜子 that arouse to the following associations: 乔治 /Qiao-Zhi/ - male given name, English origin; 布什 /Bu-Shi/ - surname, English origin; 山本 /Shan-Ben/ - surname, Japanese origin; 亚喜子 /Ya-Xi-Zi/ - female given name, Japanese origin.

In Section 2, we summarize the related work. In Section 3, we discuss the linguistic feasibility of semantic transliteration for personal names. Section 4 formulates a probabilistic model for semantic transliteration. Section 5 reports the experiments. Finally, we conclude in Section 6.

2 Related Work

In general, computational studies of transliteration fall into two categories: transliteration modeling and extraction of transliteration pairs. In transliteration modeling, transliteration rules are trained from a large, bilingual transliteration lexicon (Lin and Chen, 2002; Oh and Choi, 2005), with the objective of translating unknown words on the fly in an open, general domain. In the extraction of transliterations, data-driven methods are adopted to extract actual transliteration pairs from a corpus, in an effort to construct a large, up-to-date transliteration lexicon (Kuo et al., 2006; Sproat et al., 2006).

Phonetic transliteration can be considered as an extension to the traditional grapheme-to-phoneme (G2P) conversion (Galescu and Allen, 2001), which has been a much-researched topic in the field of speech processing. If we view the grapheme and phoneme as two symbolic representations of the same word in two different *languages*, then G2P is a transliteration task by itself. Although G2P and phonetic transliteration are common in many ways, transliteration has its unique challenges, especially as far as E-C transliteration is concerned. E-C transliteration is the conversion between English graphemes, phonetically associated English letters, and Chinese graphemes, characters which represent ideas or meanings. As a Chinese transliteration can arouse to certain connotations, the choice of Chinese characters becomes a topic of interest (Xu et al., 2006).

Semantic transliteration can be seen as a subtask of statistical machine translation (SMT) with

[3] In the literature (Knight and Graehl,1998; Qu et al., 2003), translating romanized Japanese or Chinese names to Chinese characters is also known as *back-transliteration*. For simplicity, we consider all conversions from Latin-scripted words to Chinese as transliteration in this paper.

[4] The 19 most common surnames cover 55.6% percent of the Chinese population (Ning and Ning 1995).

monotonic word ordering. By treating a letter/character as a word and a group of letters/characters as a phrase or token unit in SMT, one can easily apply the traditional SMT models, such as the IBM generative model (Brown et al., 1993) or the phrase-based translation model (Crego et al., 2005) to transliteration. In transliteration, we face similar issues as in SMT, such as lexical mapping and alignment. However, transliteration is also different from general SMT in many ways. Unlike SMT where we aim at optimizing the semantic transfer, semantic transliteration needs to maintain the phonetic equivalence as well.

In computational linguistic literature, much effort has been devoted to phonetic transliteration, such as English-Arabic, English-Chinese (Li et al., 2004), English-Japanese (Knight and Graehl, 1998) and English-Korean. In G2P studies, Font Llitjos and Black (2001) showed how knowledge of language of origin may improve conversion accuracy. Unfortunately semantic transliteration, which is considered as a good tradition in translation practice (Hu and Xu, 2003; Hu, 2004), has not been adequately addressed computationally in the literature. Some recent work (Li et al., 2006; Xu et al., 2006) has attempted to introduce preference into a probabilistic framework for selection of Chinese characters in phonetic transliteration. However, there is neither analytical result nor semantic-motivated transliteration solution being reported.

3 Feasibility of Semantic Transliteration

A Latin-scripted personal name is written in letters, which represent the pronunciations closely, whereas each Chinese character represents not only the syllables, but also the semantic associations. Thus, character rendering is a vital issue in transliteration. Good transliteration adequately projects semantic association while an inappropriate one may lead to undesirable interpretation.

Is semantic transliteration possible? Let's first conduct an inquiry into the feasibility of semantic transliteration on 3 bilingual name corpora, which are summarizied in Table 1 and will be used in experiments. E-C corpus is an augmented version of *Xinhua* English to Chinese dictionary –for English names (Xinhua, 1992). J-C corpus is a romanized Japanese to Chinese dictionary for Japanese names. The C-C corpus is a Chinese *Pinyin* to character dictionary for Chinese names. The entries are classified into surname, male and female given name categories. The E-C corpus also contains some entries without gender/surname labels, referred to as *unclassified*.

	E-C	J-C[5]	C-C[6]
Surname (S)	12,490	36,352	569,403
Given name (M)	3,201	35,767	345,044
Given name (F)	4,275	11,817	122,772
Unclassified	22,562	-	-
All	42,528	83,936	1,972,851

Table 1: Number of entries in 3 corpora

Phonetic transliteration has not been a problem as Chinese has over 400 unique syllables that are enough to approximately transcribe all syllables in other languages. Different Chinese characters may render into the same syllable and form a range of homonyms. Among the homonyms, those arousing positive meanings can be used for personal names. As discussed elsewhere (Sproat et al., 1996), out of several thousand common Chinese characters, a subset of a few hundred characters tends to be used overwhelmingly for transliterating English names to Chinese, e.g. only 731 Chinese characters are adopted in the E-C corpus. Although the character sets are shared across languages and genders, the statistics in Table 2 show that each semantic attribute is associated with some unique characters. In the C-C corpus, out of the total of 4,507 characters, only 776 of them are for surnames. It is interesting to find that female given names are represented by a smaller set of characters than that for male across 3 corpora.

	E-C	J-C	C-C	All
S	327	2,129	776	2,612 (19.2%)
M	504	1,399	4,340	4,995 (20.0%)
F	479	1,178	1,318	2,192 (26.3%)
All	731 (44.2%)	2,533 (46.2%)	4,507 (30.0%)	5,779 (53.6%)

Table 2: Chinese character usage in 3 corpora. The numbers in brackets indicate the percentage of characters that are shared by at least 2 corpora.

Note that the overlap of Chinese characters usage across genders is higher than that across languages. For instance, there is a 44.2% overlap

[5] http://www.cjk.org

[6] http://technology.chtsai.org/namelist

across gender for the transcribed English names; but only 19.2% overlap across languages for the surnames.

In summary, the semantic attributes of personal names are characterized by the choice of characters, and therefore their *n*-gram statistics as well. If the attributes are known in advance, then the semantic transliteration is absolutely feasible. We may obtain the semantic attributes from the context through trigger words. For instance, from "Mr *Tony Blair*", we realize "*Tony*" is a male given name while "*Blair*" is a surname; from "Japanese Prime Minister *Koizumi*", we resolve that "*Koizumi*" is a Japanese surname. In the case where contextual trigger words are not available, we study detecting the semantic attributes from the personal names themselves in the next section.

4 Formulation of Transliteration Model

Let *S* and *T* denote the name written in the source and target writing systems respectively. Within a probabilistic framework, a transliteration system produces the optimum target name, T^*, which yields the highest posterior probability given the source name, *S*, *i.e.*

$$T^* = \arg\max_{T \in \mathcal{T}_S} P(T \mid S) \tag{1}$$

where $\mathcal{T}_S$ is the set of all possible transliterations for the source name, *S*. The alignment between *S* and *T* is assumed implicit in the above formulation. In a standard phonetic transliteration system, $P(T \mid S)$, the posterior probability of the hypothesized transliteration, *T*, given the source name, *S*, is directly modeled without considering any form of semantic information. On the other hand, semantic transliteration described in this paper incorporates language of origin and gender information to capture the semantic structure. To do so, $P(T \mid S)$ is rewritten as

$$P(T \mid S) = \sum_{L \in \mathcal{L}, G \in \mathcal{G}} P(T, L, G \mid S) \tag{2}$$

$$= \sum_{L \in \mathcal{L}, G \in \mathcal{G}} P(T \mid S, L, G) P(L, G \mid S) \tag{3}$$

where $P(T \mid S, L, G)$ is the transliteration probability from source *S* to target *T*, given the language of origin (*L*) and gender (*G*) labels. $\mathcal{L}$ and $\mathcal{G}$ denote the sets of languages and genders respectively. $P(L, G \mid S)$ is the probability of the language and the gender given the source, *S*.

Given the alignment between *S* and *T*, the transliteration probability given *L* and *G* may be written as

$$P(T \mid S, L, G) = \prod_{i=1}^{I} P(t_i \mid T_1^{i-1}, S_1^i) \tag{4}$$

$$\approx \prod_{i=1}^{I} P(t_i \mid t_{i-1}, s_{i-1}, s_i) \tag{5}$$

where s_i and t_i are the i^{th} token of *S* and *T* respectively and *I* is the total number of tokens in both *S* and *T*. S_j^k and T_j^k represent the sequence of tokens $(s_j, s_{j+1}, \ldots, s_k)$ and $(t_j, t_{j+1}, \ldots, t_k)$ respectively. Eq. (4) is in fact the *n*-gram likelihood of the token pair $\langle t_i, s_i \rangle$ sequence and Eq. (5) approximates this probability using a bigram language model. This model is conceptually similar to the joint source-channel model (Li et al., 2004) where the target token t_i depends on not only its source token s_i but also the history t_{i-1} and s_{i-1}. Each character in the target name forms a token. To obtain the source tokens, the source and target names in the training data are aligned using the EM algorithm. This yields a set of possible source tokens and a mapping between the source and target tokens. During testing, each source name is first segmented into all possible token sequences given the token set. These source token sequences are mapped to the target sequences to yield an *N*-best list of transliteration candidates. Each candidate is scored using an *n*-gram language model given by Eqs. (4) or (5).

As in Eq. (3), the transliteration also greatly depends on the prior knowledge, $P(L, G \mid S)$. When no prior knowledge is available, a uniform probability distribution is assumed. By expressing $P(L, G \mid S)$ in the following form,

$$P(L, G \mid S) = P(G \mid L, S) P(L \mid S) \tag{6}$$

prior knowledge about language and gender may be incorporated. For example, if the language of *S* is known as L_S, we have

$$P(L \mid S) = \begin{cases} 1 & L = L_S \\ 0 & L \neq L_S \end{cases} \tag{7}$$

Similarly, if the gender information for *S* is known as G_S, then,

$$P(G \mid L,S) = \begin{cases} 1 & G = G_S \\ 0 & G \neq G_S \end{cases} \quad (8)$$

Note that personal names have clear semantic associations. In the case where the semantic attribute information is not available, we propose learning semantic information from the names themselves. Using Bayes' theorem, we have

$$P(L,G \mid S) = \frac{P(S \mid L,G)P(L,G)}{P(S)} \quad (9)$$

$P(S \mid L,G)$ can be modeled using an *n*-gram language model for the letter sequence of all the Latin-scripted names in the training set. The prior probability, $P(L,G)$, is typically uniform. $P(S)$ does not depend on *L* and *G*, thus can be omitted.

Incorporating $P(L,G \mid S)$ into Eq. (3) can be viewed as performing a soft decision of the language and gender semantic attributes. By contrast, hard decision may also be performed based on maximum likelihood approach:

$$\overline{L}_S = \arg\max_{L \in \mathcal{L}} P(S \mid L) \quad (10)$$

$$\overline{G}_S = \arg\max_{G \in \mathcal{G}} P(S \mid L,G) \quad (11)$$

where $\overline{L}_S$ and $\overline{G}_S$ are the detected language and gender of *S* respectively. Therefore, for hard decision, $P(L,G \mid S)$ is obtained by replacing L_S and G_S in Eq. (7) and (8) with $\overline{L}_S$ and $\overline{G}_S$ respectively. Although hard decision eliminates the need to compute the likelihood scores for all possible pairs of *L* and *G*, the decision errors made in the early stage will propagate to the transliteration stage. This is potentially bad if a poor detector is used (see Table 9 in Section 5.3).

If we are unable to model the prior knowledge of semantic attributes $P(L,G \mid S)$, then a more general model will be used for $P(T \mid S,L,G)$ by dropping the dependency on the information that is not available. For example, Eq. (3) is reduced to $\sum_{L \in \mathcal{L}} P(T \mid S,L)P(L \mid S)$ if the gender information is missing. Note that when both language and gender are unknown, the system simplifies to the baseline phonetic transliteration system.

5 Experiments

This section presents experiments on database of 3 language origins (Japanese, Chinese and English) and gender information (surname[7], male and female). In the experiments of determining the language origin, we used the full data set for the 3 languages as in shown in Table 1. The training and test data for semantic transliteration are the subset of Table 1 comprising those with surnames, male and female given names labels. In this paper, J, C and E stand for Japanese, Chinese and English; S, M and F represent Surname, Male and Female given names, respectively.

L	Data set	# unique entries			
		S	M	F	All
J	Train	21.7k	5.6k	1.7k	27.1k
	Test	2.6k	518	276	2.9k
C	Train	283	29.6k	9.2k	31.5k
	Test	283	2.9k	1.2k	3.1k
E	Train	12.5k	2.8k	3.8k	18.5k
	Test	1.4k	367	429	2.1k

Table 3: Number of unique entries in training and test sets, categorized by semantic attributes

Table 3 summarizes the number of *unique*[8] name entries used in training and testing. The test sets were randomly chosen such that the amount of test data is approximately 10-20% of the whole corpus. There were no overlapping entries between the training and test data. Note that the Chinese surnames are typically single characters in a small set; we assume there is no unseen surname in the test set. All the Chinese surname entries are used for both training and testing.

5.1 Language of Origin

For each language of origin, a 4-gram language model was trained for the letter sequence of the source names, with a 1-letter shift.

Japanese	Chinese	English	All
96.46	96.44	89.90	94.81

Table 4: Language detection accuracies (%) using a 4-gram language model for the letter sequence of the source name in Latin script.

[7] In this paper, surnames are treated as a special class of gender. Unlike given names, they do not have any gender association. Therefore, they fall into a third category which is neither male nor female.

[8] By contrast, Table 1 shows the total number of name examples available. For each unique entry, there may be multiple examples.

Table 4 shows the language detection accuracies for all the 3 languages using Eq. (10). The overall detection accuracy is 94.81%. The corresponding Equal Error Rate (EER)[9] is 4.52%. The detection results may be used directly to infer the semantic information for transliteration. Alternatively, the language model likelihood scores may be incorporated into the Bayesian framework to improve the transliteration performance, as described in Section 4.

5.2 Gender Association

Similarly, gender detection[10] was performed by training a 4-gram language model for the letter sequence of the source names for each language and gender pair.

Language	Male	Female	All
Japanese	90.54	80.43	87.03
Chinese	64.34	71.66	66.52
English	75.20	72.26	73.62

Table 5: Gender detection accuracies (%) using a 4-gram language model for the letter sequence of the source name in Latin script.

Table 5 summarizes the gender detection accuracies using Eq. (11) assuming language of origin is known, $\overline{G}_s = \arg\max_{G \in \mathcal{G}} P(S \mid L = L_s, G)$. The overall detection accuracies are 87.03%, 66.52% and 73.62% for Japanese, Chinese and English respectively. The corresponding EER are 13.1%, 21.8% and 19.3% respectively. Note that gender detection is generally harder than language detection. This is because the tokens (syllables) are shared very much across gender categories, while they are quite different from one language to another.

5.3 Semantic Transliteration

The performance was measured using the Mean Reciprocal Rank (MRR) metric (Kantor and Voorhees, 2000), a measure that is commonly used in information retrieval, assuming there is precisely one correct answer. Each transliteration system generated at most 50-best hypotheses for each word when computing MRR. The word and character accuracies of the top best hypotheses are also reported.

We used the phonetic transliteration system as the baseline to study the effects of semantic transliteration. The phonetic transliteration system was trained by pooling all the available training data from all the languages and genders to estimate a language model for the source-target token pairs. Table 6 compares the MRR performance of the baseline system using unigram and bigram language models for the source-target token pairs.

	J	C	E	All
Unigram	0.5109	0.4869	0.2598	0.4443
Bigram	0.5412	0.5261	0.3395	0.4895

Table 6: MRR performance of phonetic transliteration for 3 corpora using unigram and bigram language models.

The MRR performance for Japanese and Chinese is in the range of 0.48-0.55. However, due to the small amount of training and test data, the MRR performance of the English name transliteration is slightly poor (approximately 0.26-0.34). In general, a bigram language model gave an overall relative improvement of 10.2% over a unigram model.

L	G	Set	J	C	E
×	×	S	0.5366	0.7426	0.4009
		M	0.5992	0.5184	0.2875
		F	0.4750	0.4945	0.1779
		All	0.5412	0.5261	0.3395
✓	×	S	0.6500	0.7971	0.7178
		M	0.6733	0.5245	0.4978
		F	0.5956	0.5191	0.4115
		All	0.6491	0.5404	0.6228
	✓	S	0.6822	0.9969	0.7382
		M	0.7267	0.6466	0.4319
		F	0.5856	0.7844	0.4340
		All	**0.6811**	**0.7075**	**0.6294**
○	○	S	0.6541	0.6733	0.7129
		M	0.6974	0.5362	0.4821
		F	0.5743	0.6574	0.4138
		All	0.6477	0.5764	0.6168

Table 7: The effect of language and gender information on the overall MRR performance of transliteration (L=Language, G=Gender, ×=unknown, ✓=known, ○=soft decision).

Next, the scenarios with perfect language and/or gender information were considered. This com-

[9] EER is defined as the error of false acceptance and false rejection when they are equal.

[10] In most writing systems, the ordering of surname and given name is known. Therefore, gender detection is only performed for male and female classes.

parison is summarized in Table 7. All the MRR results are based on transliteration systems using bigram language models. The table clearly shows that having perfect knowledge, denoted by "✓", of language and gender helps improve the MRR performance; detecting semantic attributes using soft decision, denoted by "○", has a clear win over the baseline, denoted by "×", where semantic information is not used. The results strongly recommend the use of semantic transliteration for personal names in practice.

Next let's look into the effects of automatic language and gender detection on the performance.

	J	C	E	All
×	0.5412	0.5261	0.3395	0.4895
◇	0.6292	0.5290	0.5780	0.5734
○	**0.6162**	**0.5301**	**0.6088**	**0.5765**
✓	0.6491	0.5404	0.6228	0.5952

Table 8: The effect of language detection schemes on MRR using bigram language models and unknown gender information (hereafter, ×=unknown, ✓=known, ◇=hard decision, ○=soft decision).

Table 8 compares the MRR performance of the semantic transliteration systems with different prior information, using bigram language models. Soft decision refers to the incorporation of the language model scores into the transliteration process to improve the prior knowledge in Bayesian inference. Overall, both hard and soft decision methods gave similar MRR performance of approximately 0.5750, which was about 17.5% relatively improvement compared to the phonetic transliteration system with 0.4895 MRR. The hard decision scheme owes its surprisingly good performance to the high detection accuracies (see Table 4).

	S	M	F	All
×	0.6825	0.5422	0.5062	0.5952
◇	0.7216	0.4674	0.5162	0.5855
○	**0.7216**	**0.5473**	**0.5878**	**0.6267**
✓	0.7216	0.6368	0.6786	0.6812

Table 9: The effect of gender detection schemes on MRR using bigram language models with perfect language information.

Similarly, the effect of various gender detection methods used to obtain the prior information is shown in Table 9. The language information was assumed known *a-priori*. Due to the poorer detection accuracy for the Chinese male given names (see Table 5), hard decision of gender had led to deterioration in MRR performance of the male names compared to the case where no prior information was assumed. Soft decision of gender yielded further gains of 17.1% and 13.9% relative improvements for male and female given names respectively, over the hard decision method.

L	G	MRR	Overall Accuracy (%)	
			Word	Character
×	×	0.4895	36.87	58.39
✓	×	0.5952	46.92	65.18
	✓	0.6812	58.16	70.76
◇	◇	0.5824	47.09	66.84
○	○	**0.6122**	**49.38**	**69.21**

Table 10: Overall transliteration performance using bigram language model with various language and gender information.

Finally, Table 10 compares the performance of various semantic transliteration systems using bigram language models. The baseline phonetic transliteration system yielded 36.87% and 58.39% accuracies at word and character levels respectively; and 0.4895 MRR. It can be conjectured from the results that semantic transliteration is substantially superior to phonetic transliteration. In particular, knowing the language information improved the overall MRR performance to 0.5952; and with additional gender information, the best performance of 0.6812 was obtained. Furthermore, both hard and soft decision of semantic information improved the performance, with the latter being substantially better. Both the word and character accuracies improvements were consistent and have similar trend to that observed for MRR.

The performance of the semantic transliteration using soft decisions (last row of Table 10) achieved 25.1%, 33.9%, 18.5% relative improvement in MRR, word and character accuracies respectively over that of the phonetic transliteration (first row of Table 10). In addition, soft decision also presented 5.1%, 4.9% and 3.5% relative improvement over hard decision in MRR, word and character accuracies respectively.

5.4 Discussions

It was found that the performance of the baseline phonetic transliteration may be greatly improved by incorporating semantic information such as the language of origin and gender. Furthermore, it was found that the soft decision of language and gender

outperforms the hard decision approach. The soft decision method incorporates the semantic scores $P(L,G \mid S)$ with transliteration scores $P(T \mid S,L,G)$, involving all possible semantic specific models in the decoding process.

In this paper, there are 9 such models (3 languages $\times$ 3 genders). The hard decision relies on Eqs. (10) and (11) to decide language and gender, which only involves one semantic specific model in the decoding. Neither soft nor hard decision requires any prior information about the names. It provides substantial performance improvement over phonetic transliteration at a reasonable computational cost. If the prior semantic information is known, e.g. via trigger words, then semantic transliteration attains its best performance.

6 Conclusion

Transliteration is a difficult, artistic human endeavor, as rich as any other creative pursuit. Research on automatic transliteration has reported promising results for regular transliteration, where transliterations follow certain rules. The generative model works well as it is designed to capture regularities in terms of rules or patterns. This paper extends the research by showing that semantic transliteration of personal names is feasible and provides substantial performance gains over phonetic transliteration. This paper has presented a successful attempt towards semantic transliteration using personal name transliteration as a case study. It formulates a mathematical framework that incorporates explicit semantic information (prior knowledge), or implicit one (through soft or hard decision) into the transliteration model. Extending the framework to machine transliteration of named entities in general is a topic for further research.

References

Peter F. Brown and Stephen Della Pietra and Vincent J. Della Pietra and Robert L. Mercer. 1993, The Mathematics of Statistical Machine Translation: Parameter Estimation, *Computational Linguistics*, 19(2), pp. 263-311.

J. M. Crego, M. R. Costa-jussa and J. B. Mario and J. A. R. Fonollosa. 2005, N-gram-based versus Phrase-based Statistical Machine Translation, In *Proc. of IWSLT*, pp. 177-184.

Ariadna Font Llitjos, Alan W. Black. 2001. Knowledge of language origin improves pronunciation accuracy of proper names. In *Proc. of Eurospeech*, Denmark, pp 1919-1922.

Lucian Galescu and James F. Allen. 2001, Bi-directional Conversion between Graphemes and Phonemes using a Joint N-gram Model, In *Proc. 4th ISCA Tutorial and Research Workshop on Speech Synthesis*, Scotland, pp. 103-108.

Peter Hu, 2004, Adapting English to Chinese, *English Today*, 20(2), pp. 34-39.

Qingping Hu and Jun Xu, 2003, Semantic Transliteration: A Good Tradition in Translating Foreign Words into Chinese Babel: *International Journal of Translation, Babel*, 49(4), pp. 310-326.

Paul B. Kantor and Ellen M. Voorhees, 2000, The TREC-5 Confusion Track: Comparing Retrieval Methods for Scanned Text. *Informational Retrieval*, 2, pp. 165-176.

K. Knight and J. Graehl. 1998. Machine Transliteration, *Computational Linguistics* 24(4), pp. 599-612.

J.-S. Kuo, H. Li and Y.-K. Yang. 2006. Learning Transliteration Lexicons from the Web, In *Proc. of 44th ACL*, pp. 1129-1136.

Haizhou Li, Min Zhang and Jian Su. 2004. A Joint Source Channel Model for Machine Transliteration, In *Proc. of 42nd ACL*, pp. 159-166.

Haizhou Li, Shuanhu Bai, and Jin-Shea Kuo, 2006, *Transliteration*, In *Advances in Chinese Spoken Language Processing*, *C.-H. Lee, et al. (eds)*, World Scientific, pp. 341-364.

Wei-Hao Lin and Hsin-Hsi Chen, 2002, Backward machine transliteration by learning phonetic similarity, In *Proc. of CoNLL* , pp.139-145.

Yegao Ning and Yun Ning, 1995, *Chinese Personal Names*, Federal Publications, Singapore.

Jong-Hoon Oh and Key-Sun Choi. 2005, An Ensemble of Grapheme and Phoneme for Machine Transliteration, In *Proc. of IJCNLP*, pp.450-461.

Y. Qu, G. Grefenstette and D. A. Evans, 2003, Automatic Transliteration for Japanese-to-English Text Retrieval. In *Proc. of 26th ACM SIGIR*, pp. 353-360.

Richard Sproat, C. Chih, W. Gale, and N. Chang. 1996. A stochastic Finite-state Word-segmentation Algorithm for Chinese, Computational Linguistics, 22(3), pp. 377-404.

Richard Sproat, Tao Tao and ChengXiang Zhai. 2006. *Named Entity Transliteration with Comparable Corpora*, In *Proc. of 44th ACL*, pp. 73-80.

Xinhua News Agency, 1992, *Chinese Transliteration of Foreign Personal Names*, The Commercial Press.

L. Xu, A. Fujii, T. Ishikawa, 2006 Modeling Impression in Probabilistic Transliteration into Chinese, In *Proc. of EMNLP 2006*, Sydney, pp. 242–249.

Generating Complex Morphology for Machine Translation

Einat Minkov[*]
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA, USA
einatm@cs.cmu.edu

Kristina Toutanova
Microsoft Research
Redmond, WA, USA
kristout@microsoft.com

Hisami Suzuki
Microsoft Research
Redmond, WA, USA
hisamis@microsoft.com

Abstract

We present a novel method for predicting inflected word forms for generating morphologically rich languages in machine translation. We utilize a rich set of syntactic and morphological knowledge sources from both source and target sentences in a probabilistic model, and evaluate their contribution in generating Russian and Arabic sentences. Our results show that the proposed model substantially outperforms the commonly used baseline of a trigram target language model; in particular, the use of morphological and syntactic features leads to large gains in prediction accuracy. We also show that the proposed method is effective with a relatively small amount of data.

1 Introduction

Machine Translation (MT) quality has improved substantially in recent years due to applying data intensive statistical techniques. However, state-of-the-art approaches are essentially lexical, considering every surface word or phrase in both the source sentence and the corresponding translation as an independent entity. A shortcoming of this word-based approach is that it is sensitive to data sparsity. This is an issue of importance as aligned corpora are an expensive resource, which is not abundantly available for many language pairs. This is particularly problematic for morphologically rich languages, where word stems are realized in many different surface forms, which exacerbates the sparsity problem.

In this paper, we explore an approach in which words are represented as a collection of morphological entities, and use this information to aid in MT for morphologically rich languages. Our goal is two-fold: first, to allow generalization over morphology to alleviate the data sparsity problem in morphology generation. Second, to model syntactic coherence in the form of morphological agreement in the target language to improve the generation of morphologically rich languages. So far, this problem has been addressed in a very limited manner in MT, most typically by using a target language model.

In the framework suggested in this paper, we train a model that predicts the inflected forms of a sequence of word stems in a target sentence, given the corresponding source sentence. We use word and word alignment information, as well as lexical resources that provide morphological information about the words on both the source and target sides. Given a sentence pair, we also obtain syntactic analysis information for both the source and translated sentences. We generate the inflected forms of words in the target sentence using all of the available information, using a log-linear model that learns the relevant mapping functions.

As a case study, we focus on the English-Russian and English-Arabic language pairs. Unlike English, Russian and Arabic have very rich systems of morphology, each with distinct characteristics. Translating from a morphology-poor to a morphology-rich language is especially challenging since detailed morphological information needs to be decoded from a language that does not encode this information or does so only implicitly (Koehn, 2005). We believe that these language pairs are represen-

[*] This research was conducted during the author's internship at Microsoft Research.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 128–135,
Prague, Czech Republic, June 2007.

tative in this respect and therefore demonstrate the generality of our approach.

There are several contributions of this work. First, we propose a general approach that shows promise in addressing the challenges of MT into morphologically rich languages. We show that the use of both syntactic and morphological information improves translation quality. We also show the utility of source language information in predicting the word forms of the target language. Finally, we achieve these results with limited morphological resources and training data, suggesting that the approach is generally useful for resource-scarce language pairs.

2 Russian and Arabic Morphology

Table 1 describes the morphological features relevant to Russian and Arabic, along with their possible values. The rightmost column in the table refers to the morphological features that are shared by Russian and Arabic, including person, number, gender and tense. While these features are fairly generic (they are also present in English), note that Russian includes an additional gender (neuter) and Arabic has a distinct number notion for two (dual). A central dimension of Russian morphology is case marking, realized as suffixation on nouns and nominal modifiers[1]. The Russian case feature includes six possible values, representing the notions of subject, direct object, location, etc. In Arabic, like other Semitic languages, word surface forms may include proclitics and enclitics (or prefixes and suffixes as we refer to them in this paper), concatenated to inflected stems. For nouns, prefixes include conjunctions (*wa*: "and", *fa*: "and, so"), prepositions (*bi*: "by, with", *ka*: "like, such as", *li*: "for, to") and a determiner, and suffixes include possessive pronouns. Verbal prefixes include conjunction and negation, and suffixes include object pronouns. Both object and possessive pronouns are captured by an indicator function for its presence or absence, as well as by the features that indicate their person, number and gender. As can be observed from the table, a large number of surface inflected forms can be generated by the combination of these features, making the morphological generation of these languages a non-trivial task.

Morphologically complex languages also tend to display a rich system of agreements. In Russian, for example, adjectives agree with head nouns in number, gender and case, and verbs agree with the subject noun in person and number (past tense verbs agree in gender and number). Arabic has a similarly rich system of agreement, with unique characteristics. For example, in addition to agreement involving person, number and gender, it also requires a determiner for each word in a definite noun phrase with adjectival modifiers; in a noun compound, a determiner is attached to the last noun in the chain. Also, non-human subject plural nouns require the verb to be inflected in a singular feminine form. Generating these morphologically complex languages is therefore more difficult than generating English in terms of capturing the agreement phenomena.

3 Related Work

The use of morphological features in language modelling has been explored in the past for morphology-rich languages. For example, (Duh and Kirchhoff, 2004) showed that factored language models, which consider morphological features and use an optimized backoff policy, yield lower perplexity.

In the area of MT, there has been a large body of work attempting to modify the *input* to a translation system in order to improve the generated alignments for particular language pairs. For example, it has been shown (Lee, 2004) that determiner segmentation and deletion in Arabic sentences in an Arabic-to-English translation system improves sentence alignment, thus leading to improved overall translation quality. Another work (Koehn and Knight, 2003) showed improvements by splitting compounds in German. (Nießen and Ney, 2004) demonstrated that a similar level of alignment quality can be achieved with smaller corpora applying morpho-syntactic source restructuring, using hierarchical lexicon models, in translating from German into English. (Popović and Ney, 2004) experimented successfully with translating from inflectional languages into English making use of POS tags, word stems and suffixes in the source language. More recently, (Goldwater and McClosky, 2005) achieved improvements in Czech-English MT, optimizing a

[1]Case marking also exists in Arabic. However, in many instances, it is realized by diacritics which are ignored in standard orthography. In our experiments, we include case marking in Arabic only when it is reflected in the orthography.

Features	Russian	Arabic	Both
POS	(11 categories)	(18 categories)	
Person			1,2,3
Number		dual	sing(ular), pl(ural)
Gender	neut(er)		masc(uline), fem(inine)
Tense	gerund		present, past, future, imperative
Mood		subjunctive, jussive	
Case	dat(ive), prep(ositional), instr(umental)		nom(inative), acc(usative), gen(itive)
Negation		yes, no	
Determiner		yes, no	
Conjunction		wa, fa, none	
Preposition		bi, ka, li, none	
ObjectPronoun		yes, no	
		Pers/Numb/Gend of pronoun, none	
PossessivePronoun		Same as ObjectPronoun	

Table 1: Morphological features used for Russian and Arabic

set of possible source transformations, incorporating morphology. In general, this line of work focused on translating from morphologically rich languages into English; there has been limited research in MT in the opposite direction. Koehn (2005) includes a survey of statistical MT systems in both directions for the Europarl corpus, and points out the challenges of this task. A recent work (El-Kahlout and Oflazer, 2006) experimented with English-to-Turkish translation with limited success, suggesting that inflection generation given morphological features may give positive results.

In the current work, we suggest a probabilistic framework for morphology generation performed as *post-processing*. It can therefore be considered as complementary to the techniques described above. Our approach is general in that it is not specific to a particular language pair, and is novel in that it allows modelling of agreement on the target side. The framework suggested here is most closely related to (Suzuki and Toutanova, 2006), which uses a probabilistic model to generate Japanese case markers for English-to-Japanese MT. This work can be viewed as a generalization of (Suzuki and Toutanova, 2006) in that our model generates inflected forms of words, and is not limited to generating a small, closed set of case markers. In addition, the morphology generation problem is more challenging in that it requires handling of complex agreement phenomena along multiple morphological dimensions.

4 Inflection Prediction Framework

In this section, we define the task of of morphological generation as inflection prediction, as well as the lexical operations relevant for the task.

4.1 Morphology Analysis and Generation

Morphological analysis can be performed by applying language specific rules. These may include a full-scale morphological analysis with contextual disambiguation, or, when such resources are not available, simple heuristic rules, such as regarding the last few characters of a word as its morphogical suffix. In this work, we assume that lexicons L_S and L_T are available for the source and translation languages, respectively. Such lexicons can be created manually, or automatically from data. Given a lexicon L and a surface word w, we define the following operations:

- *Stemming* - let $S_w = \{s^1, ..., s^l\}$ be the set of possible morphological stems (lemmas) of w according to L.[2]
- *Inflection* - let $I_w = \{i^1, ..., i^m\}$ be the set of surface form words that have the same stem as w. That is, $i \in I_w$ *iff* $S_i \bigcap S_w \neq \emptyset$.
- *Morphological analysis* - let $A_w = \{a^1, ..., a^v\}$ be the set of possible morphological analyses for w. A morphological analysis a is a vector of categorical values, where the dimensions and possible values for each dimension in the vector representation space are defined by L.

4.2 The Task

We assume that we are given aligned sentence pairs, where a sentence pair includes a source and a tar-

[2]Multiple stems are possible due to ambiguity in morphological analysis.

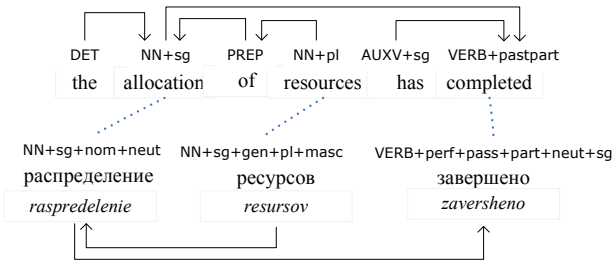


Figure 1: Aligned English-Russian sentence pair with syntactic and morphological annotation

get sentence, and lexicons L_S and L_T that support the operations described in the section above. Let a sentence $w_1, ...w_t, ...w_n$ be the output of a MT system in the target language. This sentence can be converted into the corresponding stem set sequence $S_1, ...S_t, ...S_n$, applying the stemming operation. Then the task is, for every stem set S_t in the output sentence, to predict an inflection y_t from its inflection set I_t. The predicted inflections should both reflect the meaning conveyed by the source sentence, and comply with the agreement rules of the target language. [3]

Figure 1 shows an example of an aligned English-Russian sentence pair: on the source (English) side, POS tags and word dependency structure are indicated by solid arcs. The alignments between English and Russian words are indicated by the dotted lines. The dependency structure on the Russian side, indicated by solid arcs, is given by a treelet MT system in our case (see Section 6.1), projected from the word dependency structure of English and word alignment information. Note that the Russian sentence displays agreement in number and gender between the subject noun (*raspredelenie*) and the predicate (*zaversheno*); note also that *resursov* is in genitive case, as it modifies the noun on its left.

5 Models for Inflection Prediction

5.1 A Probabilistic Model

Our learning framework uses a Maximum Entropy Markov model (McCallum et al., 2000). The model decomposes the overall probability of a predicted inflection sequence into a product of local probabilities for individual word predictions. The local probabilities are conditioned on the previous k predictions. The model implemented here is of second order: at any decision point t we condition the probability distribution over labels on the previous two predictions y_{t-1} and y_{t-2} in addition to the given (static) word context from both the source and target sentences. That is, the probability of a predicted inflection sequence is defined as follows:

$$p(\overline{y} \mid \overline{x}) = \prod_{t=1}^{n} p(y_t \mid y_{t-1}, y_{t-2}, x_t), y_t \in I_t$$

where x_t denotes the given context at position t and I_t is the set of inflections corresponding to S_t, from which the model should choose y_t.

The features we constructed pair up predicates on the *context* ($\bar{x}, y_{t-1}, y_{t-2}$) and the *target label* (y_t). In the suggested framework, it is straightforward to encode the morphological properties of a word, in addition to its surface inflected form. For example, for a particular inflected word form y_t and its context, the derived paired features may include:

$$\phi_k = \begin{cases} 1 & \text{if surface word } y_t \text{ is } y' \text{ and } s' \in S_{t+1} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_{k+1} = \begin{cases} 1 & \text{if } \textit{Gender}(y_t) = \textit{“Fem”} \text{ and } \textit{Gender}(y_{t-1}) = \textit{“Fem”} \\ 0 & \text{otherwise} \end{cases}$$

In the first example, a given neighboring stem set S_{t+1} is used as a context feature for predicting the target word y_t. The second feature captures the gender agreement with the previous word. This is possible because our model is of second order. Thus, we can derive context features describing the morphological properties of the two previous predictions.[4] Note that our model is not a simple multi-class classifier, because our features are shared across multiple target labels. For example, the gender feature above applies to many different inflected forms. Therefore, it is a structured prediction model, where the structure is defined by the morphological properties of the target predictions, in addition to the word sequence decomposition.

5.2 Feature Categories

The information available for estimating the distribution over y_t can be split into several categories,

[3] That is, assuming that the stem sequence that is output by the MT system is correct.

[4] Note that while we decompose the prediction task left-to-right, an appealing alternative is to define a top-down decomposition, traversing the dependency tree of the sentence. However, this requires syntactic analysis of sufficient quality.

corresponding to feature source. The first major distinction is monolingual versus bilingual features: *monolingual* features refer only to the context (and predicted label) in the target language, while *bilingual* features have access to information in the source sentences, obtained by traversing the word alignment links from target words to a (set of) source words, as shown in Figure 1.

Both monolingual and bilingual features can be further split into three classes: *lexical*, *morphological* and *syntactic*. *Lexical* features refer to surface word forms, as well as their stems. Since our model is of second order, our monolingual lexical features include the features of a standard word trigram language model. Furthermore, since our model is discriminative (predicting word forms given their stems), the monolingual lexical model can use stems in addition to predicted words for the left and current position, as well as stems from the *right* context. *Morphological* features are those that refer to the features given in Table 1. Morphological information is used in describing the target label as well as its context, and is intended to capture morphological generalizations. Finally, *syntactic* features can make use of syntactic analyses of the source and target sentences. Such analyses may be derived for the target language, using the pre-stemmed sentence. Without loss of generality, we will use here a dependency parsing paradigm. Given a syntactic analysis, one can construct syntactic features; for example, the stem of the *parent* word of y_t. Syntactic features are expected to be useful in capturing agreement phenomena.

5.3 Features

Table 2 gives the full set of suggested features for Russian and Arabic, detailed by type. For *monolingual lexical* features, we consider the stems of the predicted word and its immediately adjacent words, in addition to traditional word bigram and trigram features. For *monolingual morphological* features, we consider the morphological attributes of the two previously predicted words and the current prediction; for *monolingual syntactic* features, we use the stem of the parent node.

The bilingual features include the set of words aligned to the focus word at position t, where they are treated as bag-of-words, i.e., each aligned word is assigned a separate feature. *Bilingual lexical* features can refer to words aligned to y_t as all as words aligned to its immediate neighbors y_{t-1} and y_{t+1}. *Bilingual morphological and syntactic* features refer to the features of the source language, which are expected to be useful for predicting morphology in the target language. For example, the bilingual *Det* (determiner) feature is computed according to the source dependency tree: if a child of a word aligned to w_t is a determiner, then the feature value is assigned its surface word form (such as *a* or *the*). The bilingual *Prep* feature is computed similarly, by checking the parent chain of the word aligned to w_t for the existence of a preposition. This feature is hoped to be useful for predicting Arabic inflected forms with a prepositional prefix, as well as for predicting case marking in Russian. The bilingual *ObjPron* and *PossPron* features represent any object pronoun of the word aligned to w_t and a preceding possessive pronoun, respectively. These features are expected to map to the object and possessive pronoun features in Arabic. Finally, the bilingual *Compound* feature checks whether a word appears as part of a noun compound in the English source. f this is the case, the feature is assigned the value of "head" or "dependent". This feature is relevant for predicting a genitive case in Russian and definiteness in Arabic.

Feature categories	**Instantiations**
Monolingual lexical	
Word stem	s_{t-1},s_{t-2},s_t,s_{t+1}
Predicted word	y_t, y_{t-1}, y_{t-2}
Monolingual morphological	
f : POS, Person, Number, Gender, Tense Neg, Det, Prep, Conj, ObjPron, PossPron	$f(y_{t-2})$,$f(y_{t-1})$,$f(y_t)$
Monolingual syntactic	
Parent stem	$s_{HEAD(t)}$
Bilingual lexical	
Aligned word set Al	Al_t, Al_{t-1}, Al_{t+1}
Bilingual morph & syntactic	
f : POS, Person, Number, Gender, Tense Neg, Det, Prep, Conj, ObjPron, PossPron, Comp	$f(Al_t)$, $f(Al_{t-1})$, $f(Al_{t+1})$, $f(Al_{HEAD(t)})$

Table 2: The feature set suggested for English-Russian and English-Arabic pairs

6 Experimental Settings

In order to evaluate the effectiveness of the suggested approach, we performed *reference experiments*, that is, using the aligned sentence pairs of

Data	Eng-Rus		Eng-Ara	
Avg. sentlen	**Eng**	**Rus**	**Eng**	**Ara**
Training	1M		470K	
	14.06	12.90	12.85	11.90
Development	1,000		1,000	
	13.73	12.91	13.48	12.90
Test	1,000		1,000	
	13.61	12.84	8.49	7.50

Table 3: Data set statistics: corpus size and average sentence length (in words)

reference translations rather than the output of an MT system as input.[5] This allows us to evaluate our method with a reduced noise level, as the words and word order are perfect in reference translations. These experiments thus constitute a preliminary step for tackling the real task of inflecting words in MT.

6.1 Data

We used a corpus of approximately 1 million aligned sentence pairs for English-Russian, and 0.5 million pairs for English-Arabic. Both corpora are from a technical (software manual) domain, which we believe is somewhat restricted along some morphological dimensions, such as tense and person. We used 1,000 sentence pairs each for development and testing for both language pairs. The details of the datasets used are given in Table 3.

The sentence pairs were word-aligned using GIZA++ (Och and Ney, 2000) and submitted to a treelet-based MT system (Quirk et al., 2005), which uses the word dependency structure of the source language and projects word dependency structure to the target language, creating the structure shown in Figure 1 above.

6.2 Lexicon

Table 4 gives some relevant statistics of the lexicons we used. For Russian, a general-domain lexicon was available to us, consisting of about 80,000 lemmas (stems) and 9.4 inflected forms per stem.[6] Limiting the lexicon to word types that are seen in the training set reduces its size substantially to about 14,000 stems, and an average of 3.8 inflections per stem. We will use this latter "domain-adapted" lexicon in our experiments.

[5] In this case, y_t should equal w_t, according to the task definition.

[6] The averages reported in Table 4 are by type and do not consider word frequencies in the data.

	Source	Stems	Avg($\mid I \mid$)	Avg($\mid S \mid$)
Rus.	Lexicon	79,309	9.4	
	Lexicon $\cap$ Train	13,929	3.8	1.6
Ara.	Lexicon $\cap$ Train	12,670	7.0	1.7

Table 4: Lexicon statistics

For Arabic, as a full-size Arabic lexicon was not available to us, we used the Buckwalter morphological analyzer (Buckwalter, 2004) to derive a lexicon. To acquire the *stemming* and *inflection* operators, we submit all words in our training data to the Buckwalter analyzer. Note that Arabic displays a high level of ambiguity, each word corresponding to many possible segmentations and morphological analyses; we considered all of the different stems returned by the Buckwalter analyzer in creating a word's stem set. The lexicon created in this manner contains 12,670 distinct stems and 89,360 inflected forms.

For the generation of *word features*, we only consider one dominant analysis for any surface word for simplicity. In case of ambiguity, we considered only the first (arbitrary) analysis for Russian. For Arabic, we apply the following heuristic: use the most frequent analysis estimated from the gold standard labels in the Arabic Treebank (Maamouri et al., 2005); if a word does not appear in the treebank, we choose the first analysis returned by the Buckwalter analyzer. Ideally, the best word analysis should be provided as a result of contextual disambiguation (e.g., (Habash and Rambow, 2005)); we leave this for future work.

6.3 Baseline

As a baseline, we pick a morphological inflection y_t at random from I_t. This random baseline serves as an indicator for the difficulty of the problem. Another more competitive baseline we implemented is a word trigram language model (LM). The LMs were trained using the CMU language modelling toolkit (Clarkson and Rosenfeld, 1997) with default settings on the training data described in Table 3.

6.4 Experiments

In the experiments, our primary goal is to evaluate the effectiveness of the proposed model using all features available to us. Additionally, we are interested in knowing the contribution of each information source, namely of morpho-syntactic and bilingual features. Therefore, we study the performance

of models including the full feature schemata as well as models that are restricted to feature subsets according to the feature types as described in Section 5.2. The models are as follows: *Monolingual-Word*, including LM-like and stem n-gram features only; *Bilingual-Word*, which also includes bilingual lexical features;[7] *Monolingual-All*, which has access to all the information available in the target language, including morphological and syntactic features; and finally, *Bilingual-All*, which includes all feature types from Table 2.

For each model and language, we perform feature selection in the following manner. The features are represented as feature *templates*, such as "POS=X", which generate a set of binary features corresponding to different instantiations of the template, as in "POS=NOUN". In addition to individual features, conjunctions of up to three features are also considered for selection (e.g., "POS=NOUN & Number=plural"). Every conjunction of feature templates considered contains at least one predicate on the prediction y_t, and up to two predicates on the context. The feature selection algorithm performs a greedy forward stepwise feature selection on the feature templates so as to maximize development set accuracy. The algorithm is similar to the one described in (Toutanova, 2006). After this process, we performed some manual inspection of the selected templates, and finally obtained 11 and 36 templates for the *Monolingual-All* and *Bilingual-All* settings for Russian, respectively. These templates generated 7.9 million and 9.3 million binary feature instantiations in the final model, respectively. The corresponding numbers for Arabic were 27 feature templates (0.7 million binary instantiations) and 39 feature templates (2.3 million binary instantiations) for *Monolingual-All* and *Bilingual-All*, respectively.

7 Results and Discussion

Table 5 shows the accuracy of predicting word forms for the baseline and proposed models. We report accuracy only on words that appear in our lexicons. Thus, punctuation, English words occurring in the target sentence, and words with unknown lemmas are excluded from the evaluation. The reported accuracy measure therefore abstracts away from the issue of incomplete coverage of the lexicon. When we encounter these words in the true MT scenario, we will make no predictions about them, and simply leave them unmodified. In our current experiments, in Russian, 68.2% of all word tokens were in Cyrillic, of which 93.8% were included in our lexicon. In Arabic, 85.5% of all word tokens were in Arabic characters, of which 99.1% were in our lexicon.[8]

Model	**Eng-Rus**	**Eng-Ara**
Random	31.7	16.3
LM	77.6	31.7
Monolingual Word	85.1	69.6
Bilingual Word	87.1	71.9
Monolingual All	87.1	71.6
Bilingual All	**91.5**	**73.3**

Table 5: Accuracy (%) results by model

The results in Table 5 show that the suggested models outperform the language model substantially for both languages. In particular, the contribution of both bilingual and non-lexical features is noteworthy: adding non-lexical features consistently leads to 1.5% to 2% absolute gain in both monolingual and bilingual settings in both language pairs. We obtain a particularly large gain in the Russian bilingual case, in which the absolute gain is more than 4%, translating to 34% error rate reduction. Adding bilingual features has a similar effect of gaining about 2% (and 4% for Russian non-lexical) in accuracy over monolingual models. The overall accuracy is lower in Arabic than in Russian, reflecting the inherent difficulty of the task, as indicated by the random baseline (31.7 in Russian vs. 16.3 in Arabic).

In order to evaluate the effectiveness of the model in alleviating the data sparsity problem in morphological generation, we trained inflection prediction models on various subsets of the training data described in Table 3, and tested their accuracy. The results are given in Figure 2. We can see that with as few as 5,000 training sentences pairs, the model obtains much better accuracy than the language model, which is trained on data that is larger by a few orders of magnitude. We also note that the learning curve

[7] Overall, this feature set approximates the information that is available to a state-of-the-art statistical MT system.

[8] For Arabic, the inflection ambiguity was extremely high: there were on average 39 inflected forms per stem set in our development corpus (per token), as opposed to 7 in Russian. We therefore limited the evaluation of Arabic to those stems that have up to 30 inflected forms, resulting in 17 inflected forms per stem set on average in the development data.

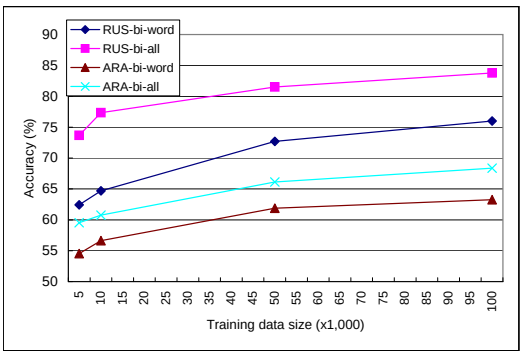


Figure 2: Accuracy, varying training data size

becomes less steep as we use more training data, suggesting that the models are successfully learning generalizations.

We have also manually examined some representative cases where the proposed model failed to make a correct prediction. In both Russian and Arabic, a very common pattern was a mistake in predicting the gender (as well as number and person in Arabic) of pronouns. This may be attributed to the fact that the correct choice of the pronoun requires coreference resolution, which is not available in our model. A more thorough analysis of the results will be helpful to bring further improvements.

8 Conclusions and Future Work

We presented a probabilistic framework for morphological generation given aligned sentence pairs, incorporating morpho-syntactic information from both the source and target sentences. The results, using reference translations, show that the proposed models achieve substantially better accuracy than language models, even with a relatively small amount of training data. Our models using morpho-syntactic information also outperformed models using only lexical information by a wide margin. This result is very promising for achieving our ultimate goal of improving MT output by using a specialized model for target language morphological generation. Though this goal is clearly outside the scope of this paper, we conducted a preliminary experiment where an English-to-Russian MT system was trained on a stemmed version of the aligned data and used to generate stemmed word sequences, which were then inflected using the suggested framework. This simple integration of the proposed model with the MT system improved the BLEU score by 1.7. The most obvious next step of our research, therefore, is to further pursue the integration of the proposed model to the end-to-end MT scenario.

There are multiple paths for obtaining further improvements over the results presented here. These include refinement in feature design, word analysis disambiguation, morphological and syntactic analysis on the source English side (e.g., assigning semantic role tags), to name a few. Another area of investigation is capturing longer-distance agreement phenomena, which can be done by implementing a global statistical model, or by using features from dependency trees more effectively.

References

Tim Buckwalter. 2004. Buckwalter arabic morphological analyzer version 2.0.

Philip Clarkson and Roni Rosenfeld. 1997. Statistical language modelling using the CMU cambridge toolkit. In *Eurospeech*.

Kevin Duh and Kathrin Kirchhoff. 2004. Automatic learning of language model structure. In *COLING*.

Ilknur Durgar El-Kahlout and Kemal Oflazer. 2006. Initial explorations in English to Turkish statistical machine translation. In *NAACL workshop on statistical machine translation*.

Sharon Goldwater and David McClosky. 2005. Improving statistical MT through morphological analysis. In *EMNLP*.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *ACL*.

Philipp Koehn and Kevin Knight. 2003. Empirical methods for compound splitting. In *EACL*.

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

Young-Suk Lee. 2004. Morphological analysis for statistical machine translation. In *HLT-NAACL*.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Hubert Jin. 2005. *Arabic Treebank: Part 1 v 3.0*. Linguistic Data Consortium.

Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. 2000. Maximum entropy markov models for information extraction and segmentation. In *ICML*.

Sonja Nießen and Hermann Ney. 2004. Statistical machine translation with scarce resources using morpho-syntactic information. *Computational Linguistics*, 30(2):181–204.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *ACL*.

Maja Popović and Hermann Ney. 2004. Towards the use of word stems and suffixes for statistical machine translation. In *LREC*.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency tree translation: Syntactically informed phrasal SMT. In *ACL*.

Hisami Suzuki and Kristina Toutanova. 2006. Learning to predict case markers in Japanese. In *COLING-ACL*.

Kristina Toutanova. 2006. Competitive generative models with structure learning for NLP classification tasks. In *EMNLP*.

Assisting Translators in Indirect Lexical Transfer

Bogdan Babych, Anthony Hartley, Serge Sharoff
Centre for Translation Studies
University of Leeds, UK
{b.babych,a.hartley,s.sharoff}@leeds.ac.uk

Olga Mudraya
Department of Linguistics
Lancaster University, UK
o.mudraya@lancs.ac.uk

Abstract

We present the design and evaluation of a *translator's amenuensis* that uses comparable corpora to propose and rank non-literal solutions to the translation of expressions from the general lexicon. Using distributional similarity and bilingual dictionaries, the method outperforms established techniques for extracting translation equivalents from parallel corpora. The interface to the system is available at: http://corpus.leeds.ac.uk/assist/v05/

1 Introduction

This paper describes a system designed to assist humans in translating expressions that do not necessarily have a literal or compositional equivalent in the target language (TL). In the spirit of (Kay, 1997), it is intended as a *translator's amenuensis* "under the tight control of a human translator … to help increase his productivity and not to supplant him".

One area where human translators particularly appreciate assistance is in the translation of expressions from the general lexicon. Unlike equivalent technical terms, which generally share the same part-of-speech (POS) across languages and are in the ideal case univocal, the contextually appropriate equivalents of general language expressions are often indirect and open to variation. While the transfer module in RBMT may acceptably undergenerate through a many-to-one mapping between source and target expressions, human translators, even in non-literary fields, value legitimate variation. Thus the French expression *il faillit échouer* (lit.: he faltered to fail) may be variously rendered as *he almost/nearly/all but failed; he was on the verge/brink of failing/failure; failure loomed*. All of these translations are indirect in that they involve lexical shifts or POS transformations.

Finding such translations is a hard task that can benefit from automated assistance. 'Mining' such indirect equivalents is difficult, precisely because of the structural mismatch, but also because of the paucity of suitable aligned corpora. The approach adopted here includes the use of comparable corpora in source and target languages, which are relatively easy to create. The challenge is to generate a list of usable solutions and to rank them such that the best are at the top.

Thus the present system is unlike SMT (Och and Ney, 2003), where lexical selection is effected by a translation model based on aligned, parallel corpora, but the novel techniques it has developed are exploitable in the SMT paradigm. It also differs from now traditional uses of comparable corpora for detecting translation equivalents (Rapp, 1999) or extracting terminology (Grefenstette, 2002), which allows a one-to-one correspondence irrespective of the context. Our system addresses difficulties in expressions in the general lexicon, whose translation is context-dependent.

The structure of the paper is as follows. In Section 2 we present the method we use for mining translation equivalents. In Section 3 we present the results of an objective evaluation of the quality of suggestions produced by the system by comparing our output against a parallel corpus. Finally, in Section 4 we present a subjective evaluation focusing on the integration of the system into the workflow of human translators.

2 Methodology

The software acts as a decision support system for translators. It integrates different technologies for

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 136–143,
Prague, Czech Republic, June 2007.

extracting indirect translation equivalents from large comparable corpora. In the following subsections we give the user perspective on the system and describe the methodology underlying each of its sub-tasks.

2.1 User perspective

Unlike traditional dictionaries, the system is a *dynamic translation resource* in that it can successfully find translation equivalents for units which have not been stored in advance, even for idiosyncratic multiword expressions which almost certainly will not figure in a dictionary. While our system can rectify gaps and omissions in static lexicographical resources, its major advantage is that it is able to cope with an open set of translation problems, searching for translation equivalents in comparable corpora in runtime. This makes it more than just an extended dictionary.

Contextual descriptors

From the user perspective the system extracts indirect translation equivalents as sets of *contextual descriptors* – content words that are lexically central in a given sentence, phrase or construction. The choice of these descriptors may determine the general syntactic perspective of the sentence and the use of supporting lexical items. Many translation problems arise from the fact that the mapping between such descriptors is not straightforward.

The system is designed to find possible indirect mappings between sets of descriptors and to verify the acceptability of the mapping into the TL. For example, in the following Russian sentence, the bolded contextual descriptors require indirect translation into English.

Дети посещают ***плохо отремонтированные*** *школы, в которых* ***недостает*** *самого* ***необходимого***
(Children attend ***badly repaired*** *schools, in which [it] is* ***missing*** *the most* ***necessary)***

Combining direct translation equivalents of these words (e.g., translations found in the Oxford Russian Dictionary – ORD) may produce a non-natural English sentence, like the literal translation given above. In such cases human translators usually apply structural and lexical transformations, for instance changing the descriptors' POS and/or replacing them with near-synonyms which fit together in the context of a TL sentence (Munday, 2001: 57-58). Thus, a structural transformation of *плохо отремонтированные* (badly repaired) may give *in poor repair* while a lexical transformation of *недостает самого необходимого* ([it] is missing the most necessary) gives *lacking basic essentials*.

Our system models such transformations of the descriptors and checks the consistency of the resulting sets in the TL.

Using the system

Human translators submit queries in the form of one or more SL descriptors which in their opinion may require indirect translation. When the translators use the system for translating into their native language, the returned descriptors are usually sufficient for them to produce a correct TL construction or phrase around them (even though the descriptors do not always form a naturally sounding expression). When the translators work into a non-native language, they often find it useful to generate concordances for the returned descriptors to verify their usage within TL constructions.

For example, for the sentence above translators may submit two queries: *плохо отремонтированные* (badly repaired) and *недостает необходимого* (missing necessary). For the first query the system returns a list of descriptor pairs (with information on their frequency in the English corpus) ranked by distributional proximity to the original query, which we explain in Section 2.2. At the top of the list come:

```
bad repair = 30        (11.005)
bad maintenance = 16   (5.301)
bad restoration = 2    (5.079)
poor repair = 60       (5.026)…
```

Underlined hyperlinks lead translators to actual contexts in the English corpus, e.g., *poor repair* generates a concordance containing a desirable TL construction which is a structural transformation of the SL query:

in such a	**poor**	state of **repair**
bridge in as	**poor**	a state of **repair** as the highways
building in	**poor**	**repair**.
dwellings are in	**poor**	**repair**;

Similarly, the result of the second query may give the translators an idea about possible lexical transformation:

```
missing need = 14      (5.035)
important missing = 8  (2.930)
missing vital = 8      (2.322)
lack necessary = 204   (1.982)…
essential lack = 86    (0.908)…
```

The concordance for the last pair of descriptors contains the phrase *they lack the three essentials*, which illustrates the transformation. The resulting translation may be the following:

*Children attend schools that are in **poor repair** and **lacking** basic **essentials***

Thus our system supports translators in making decisions about indirect translation equivalents in a number of ways: it suggests possible structural and lexical transformations for contextual descriptors; it verifies which translation variants co-occur in the TL corpus; and it illustrates the use of the transformed TL lexical descriptors in actual contexts.

2.2 Generating translation equivalents

We have generalised the method used in our previous study (Sharoff et al., 2006) for extracting equivalents for continuous multiword expressions (MWEs). Essentially, the method expands the search space for each word and its dictionary translations with entries from automatically computed thesauri, and then checks which combinations are possible in target corpora. These potential translation equivalents are then ranked by their similarity to the original query and presented to the user. The range of retrievable equivalents is now extended from a relatively limited range of two-word constructions which mirror POS categories in SL and TL to a much wider set of co-occurring lexical content items, which may appear in a different order, at some distance from each other, and belong to different POS categories.

The method works best for expressions from the general lexicon, which do not have established equivalents, but not yet for terminology. It relies on a high-quality bilingual dictionary (en-ru ~30k, ru-en ~50K words, combining ORD and the core part of Multitran) and large comparable corpora (~200M En, ~70M Ru) of news texts.

For each of the SL query terms *q* the system generates its *dictionary translation Tr(q)* and its *similarity class S(q)* – a set of words with a similar distribution in a monolingual corpus. Similarity is measured as the cosine between collocation vectors, whose dimensionality is reduced by SVD using the implementation by Rapp (2004). The descriptor and each word in the similarity class are then translated into the TL using ORD or the Multitran dictionary, resulting in $\{Tr(q) \cup Tr(S(q))\}$. On the TL side we also generate similarity classes, but only for dictionary translations of query terms *Tr(q)* (not for *Tr(S(q))*, which can make output too noisy). We refer to the resulting set of TL words as a *translation class T*.

$T = \{Tr(q) \cup Tr(S(q)) \cup S(Tr(q))\}$

Translation classes approximate lexical and structural transformations which can potentially be applied to each of the query terms. Automatically computed similarity classes do not require resources like WordNet, and they are much more suitable for modelling translation transformations, since they often contain a wider range of words of different POS which share the same context, e.g., the similarity class of the word *lack* contains words such as *absence, insufficient, inadequate, lost, shortage, failure, paucity, poor, weakness, inability, need.* This clearly goes beyond the range of traditional thesauri.

For multiword queries, the system performs a consistency check on possible combinations of words from different translation classes. In particular, it computes the Cartesian product for pairs of translation classes T_1 and T_2 to generate the set P of word pairs, where each word (w_1 and w_2) comes from a different translation class:

$P = T_1 \times T_2 = \{(w_1, w_2) \mid w_1 \in T_1 \text{ and } w_2 \in T_2\}$

Then the system checks whether each word pair from the set P exists in the database D of discontinuous content word bi-grams which actually co-occur in the TL corpus:

$P' = P \cap D$

The database contains the set of all bi-grams that occur in the corpus with a frequency ≥ 4 within a window of 5 words (over 9M bigrams for each language). The bi-grams in D and in P are sorted alphabetically, so their order in the query is not important.

Larger N-grams ($N > 2$) in queries are split into combinations of bi-grams, which we found to be an optimal solution to the problem of the scarcity of higher order N-grams in the corpus. Thus, for the query *gain significant importance* the system generates $P'_{1(\text{significant importance})}$, $P'_{2(\text{gain importance})}$, $P'_{3(\text{gain significant})}$ and computes P' as:

$P' = \{(w_1,w_2,w_3) \mid (w_1,w_2) \in P'_1 \ \& \ (w_1, w_3) \in P'_2 \ \& \ (w_2,w_3) \in P'_3\},$

which allows the system to find an indirect equivalent *получить весомое значение* (lit.: receive weighty meaning).

Even though *P'* on average contains about 2% - 4% of the theoretically possible number of bigrams present in *P*, the returned number of potential translation equivalents may still be large and contain much noise. Typically there are several hundred elements in *P'*, of which only a few are really useful for translation. To make the system usable in practice, i.e., to get useful solutions to appear close to the top (preferably on the first screen of the output), we developed methods of ranking and filtering the returned TL contextual descriptor pairs, which we present in the following sections.

2.3 Hypothesis ranking

The system ranks the returned list of contextual descriptors by their distributional proximity to the original query, i.e. it uses scores $cos(v_q, v_w)$ generated for words in similarity classes – the cosine of the angle between the collocation vector for a word and the collocation vector for the query or dictionary translation of the query. Thus, words whose equivalents show similar usage in a comparable corpus receive the highest scores. These scores are computed for each individual word in the output, so there are several ways to combine them to weight words in translation classes and word combinations in the returned list of descriptors.

We established experimentally that the best way to combine similarity scores is to multiply weights *W(T)* computed for each word within its translation class *T*. The weight $W(P'_{(w1,w2)})$ for each pair of contextual descriptors $(w_1, w_2) \in P'$ is computed as:

$$W(P'_{(w1,w2)}) = W(T_{(w1)}) \times W(T_{(w2)});$$

Computing $W(T_{(w)})$, however, is not straightforward either, since some words in similarity classes of different translation equivalents for the query term may be the same, or different words from the similarity class of the original query may have the same translation. Therefore, a word *w* within a translation class may have come by several routes simultaneously, and may have done that several times. For each word *w* in *T* there is a possibility that it arrived in T either because it is in *Tr(q)* or occurs *n* times in *Tr(S(q))* or *k* times in *S(Tr(q))*.

We found that the number of occurrences *n* and *k* of each word *w* in each subset gives valuable information for ranking translation candidates. In our experiments we computed the weight *W(T)* as the sum of similarity scores which *w* receives in each of the subsets. We also discovered that ranking improves if for each query term we compute in addition a larger (and potentially noisy) space of candidates that includes TL similarity classes of translations of the SL similarity class *S(Tr(S(q)))*. These candidates do not appear in the system output, but they play an important role in ranking the displayed candidates. The improvement may be due to the fact that this space is much larger, and may better support relevant candidates since there is a greater chance that appropriate indirect equivalents are found several times within SL and TL similarity classes. The best ranking results were achieved when the original *W(T)* scores were multiplied by 2 and added to the scores for the newly introduced similarity space *S(Tr(S(q)))*:

$$W(T_{(w)}) = 2 \times (1 \text{ if } w \in Tr(q)) + 2 \times \sum(cos(v_q, v_{Tr(w)}) \mid \{w \mid w \in Tr(S(q))\}) + 2 \times \sum(cos(v_{Tr(q)}, v_w) \mid \{w \mid w \in S(Tr(q))\}) + \sum(cos(v_q, v_{Tr(w)}) \times cos(v_{Tr(q)}, v_w) \mid \{w \mid w \in S(Tr(S(q)))\})$$

For example, the system gives the following ranking for the indirect translation equivalents of the Russian phrase *весомое значение* (lit.: weighty meaning) – figures in brackets represent *W(P')* scores for each pair of TL descriptors:

```
1. significant importance = 7 (3.610)
2. significant value = 128    (3.211)
3. measurable value = 6       (2.657)…
8. dramatic importance = 2    (2.028)
9. important significant = 70 (2.014)
10. convincing importance = 6 (1.843)
```

The Russian similarity class for *весомый* (weighty, ponderous) contains: *убедительный* (convincing) (0.469), *значимый* (significant) (0.461), *ощутимый* (notable) (0.452) *драматичный* (dramatic) (0.371). The equivalent of *significant* is not at the top of the similarity class of the Russian query, but it appears at the top of the final ranking of pairs in *P'*, because this hypothesis is supported by elements of the set formed by *S(Tr(S(q)))*; it appears in similarity classes for *notable* (0.353) and *dramatic* (0.315), which contributed these values to the *W(T)* score of *significant*:

W(T(`significant`*))* =

2 × (Tr(значимый)=`significant` (0.461))
+ (Tr(ощутимый)=`notable` (0.452)
× S(`notable`)=`significant` (0.353))
+ (Tr(драматичный)=`dramatic` (0.371)
× S(`dramatic`)= `significant` (0.315))

The word *dramatic* itself is not usable as a translation equivalent in this case, but its similarity

class contains the support for relevant candidates, so it can be viewed as useful noise. On the other hand, the word *convincing* does not receive such support from the hypothesis space, even though its Russian equivalent is ranked higher in the SL similarity class.

2.4 Semantic filtering

Ranking of translation candidates can be further improved when translators use an option to filter the returned list by certain lexical criteria, e.g., to display only those examples that contain a certain lexical item, or to require one of the items to be a dictionary translation of the query term. However, lexical filtering is often too restrictive: in many cases translators need to see a number of related words from the same semantic field or subject domain, without knowing the lexical items in advance. In this section we present the semantic filter, which is based on Russian and English semantic taggers which use the same semantic field taxonomy for both languages.

The semantic filter displays only those items which have specified semantic field tags or tag combinations; it can be applied to one or both words in each translation hypothesis in *P'*. The default setting for the semantic filter is the requirement for both words in the resulting TL candidates to contain any of the semantic field tags from a SL query term.

In the next section we present evaluation results for this default setting (which is applied when the user clicks the *Semantic Filter* button), but human translators have further options – to filter by tags of individual words, to use semantic classes from SL or TL terms, etc.

For example, applying the default semantic filter for the output of the query *плохо отремонтированные* (badly repaired) removes the highlighted items from the list:

```
 1. bad repair = 30        (11.005)
[2. good repair = 154      (8.884) ]
 3. bad rebuild = 6        (5.920)
[4. bad maintenance = 16   (5.301) ]
 5. bad restoration = 2    (5.079)
 6. poor repair = 60       (5.026)
[7. good rebuild = 38      (4.779) ]
 8. bad construction = 14  (4.779)
```

Items 2 and 7 are generated by the system because *good*, *well* and *bad* are in the same similarity cluster for many words (they often share the same collocations). The semantic filter removes examples with *good* and *well* on the grounds that they do not have any of the tags which come from the word *плохо* (badly): in particular, instead of tag `A5-` (Evaluation: Negative) they have tag `A5+` (Evaluation: Positive). Item 4 is removed on the grounds that the words *отремонтированный* (repaired) and *maintenance* do not have any tags in common – they appear ontologically too far apart from the point of view of the semantic tagger.

The core of the system's multilingual semantic tagging is a knowledge base in which single words and MWEs are mapped to their potential semantic field categories. Often a lexical item is mapped to multiple semantic categories, reflecting its potential multiple senses. In such cases, the tags are arranged by the order of likelihood of meanings, with the most prominent first.

3 Objective evaluation

In the *objective evaluation* we tested the performance of our system on a selection of indirect translation problems, extracted from a parallel corpus consisting mostly of articles from English and Russian newspapers (118,497 words in the R-E direction, 589,055 words in the E-R direction). It has been aligned on the sentence level by JAPA (Langlais et al., 1998), and further on the word level by GIZA++ (Och and Ney, 2003).

3.1 Comparative performance

The intuition behind the objective evaluation experiment is that the capacity of our tool to find indirect translation equivalents in comparable corpora can be compared with the results of automatic alignment of parallel texts used in translation models in SMT: one of the major advantages of the SMT paradigm is its ability to reuse indirect equivalents found in parallel corpora (equivalents that may never come up in hand-crafted dictionaries). Thus, automatically generated GIZA++ dictionaries with word alignment contain many examples of indirect translation equivalents.

We use these dictionaries to simulate the generator of translation classes *T*, which we recombine to construct their Cartesian product *P*, similarly to the procedure we use to generate the output of our system. However, the two approaches generate indirect translation equivalence hypotheses on the basis of radically different material: the GIZA dictionary uses evidence from parallel corpora of ex-

isting human translations, while our system recombines translation candidates on the basis of their distributional similarity in monolingual comparable corpora. Therefore we took GIZA as a baseline.

Translation problems for the objective evaluation experiment were manually extracted from two parallel corpora: a section of about 10,000 words of a corpus of English and Russian newspapers, which we also used to train GIZA, and a section of the same length from a corpus of interviews published on the Euronews.net website.

We selected expressions which represented cases of lexical transformations (as illustrated in Section 0), containing at least two content words both in the SL and TL. These expressions were converted into pairs of contextual descriptors – e.g., *recent success, reflect success* – and submitted to the system and to the GIZA dictionary. We compared the ability of our system and of GIZA to find indirect translation equivalents which matched the equivalents used by human translators. The output from both systems was checked to see whether it contained the contextual descriptors used by human translators. We submitted 388 pairs of descriptors extracted from the newspaper translation corpus and 174 pairs extracted from the Euronews interview corpus. Half of these pairs were Russian, and the other half English.

We computed recall figures for 2-word combinations of contextual descriptors and single descriptors within those combinations. We also show the recall of translation variants provided by the ORD on this data set. For example, for the query *недостает необходимого* ([it] is missing necessary [things]) human translators give the solution *lacking essentials*; the lemmatised descriptors are *lack* and *essential*. ORD returns direct translation equivalents *missing* and *necessary*. The GIZA dictionary in addition contains several translation equivalents for the second term (with alignment probabilities) including: *necessary* ~0.332*, need* ~0.226*, essential* ~0.023. Our system returns both descriptors used in human translation as a pair – *lack essential* (ranked 41 without filtering and 22 with the default semantic filter). Thus, for a 2-word combination of the descriptors only the output of our system matched the human solution, which we counted as one hit for the system and no hits for ORD or GIZA. For 1-word descriptors we counted 2 hits for our system (both words in the human solution are matched), and 1 hit for GIZA – it matches the word *essential* ~0.023 (which also illustrates its ability to find indirect translation equivalents).

	2w descriptors		**1w descriptors**	
	news	**interv**	**news**	**interv**
ORD	6.7%	4.6%	32.9%	29.3%
GIZA++	13.9%	3.4%	35.6%	29.0%
Our system	**21.9%**	**19.5%**	**55.8%**	**49.4%**

Table 1 Conservative estimate of recall

It can be seen from **Table 1** that for the newspaper corpus on which it was trained, GIZA covers a wider set of indirect translation variants than ORD. But our recall is even better both for 2-word and 1-word descriptors.

However, note that GIZA's ability to retrieve from the newspaper corpus certain indirect translation equivalents may be due to the fact that it has previously seen them frequently enough to generate a correct alignment and the corresponding dictionary entry.

The Euronews interview corpus was not used for training GIZA. It represents spoken language and is expected to contain more 'radical' transformations. The small decline in ORD figures here can be attributed to the fact that there is a difference in genre between written and spoken texts and consequently between transformation types in them. However, the performance of GIZA drops radically on unseen text and becomes approximately the same as the ORD.

This shows that indirect translation equivalents in the parallel corpus used for training GIZA are too sparse to be learnt one by one and successfully applied to unseen data, since solutions which fit one context do not necessarily suit others.

The performance of our system stays at about the same level for this new type of text; the decline in its performance is comparable to the decline in ORD figures, and can again be explained by the differences in genre.

3.2 Evaluation of hypothesis ranking

As we mentioned, correct ranking of translation candidates improves the usability of the system. Again, the objective evaluation experiment gives only a conservative estimate of ranking, because there may be many more useful indirect solutions further up the list in the output of the system which are legitimate variants of the solutions found in the

parallel corpus. Therefore, evaluation figures should be interpreted in a comparative rather then an absolute sense.

We use ranking by frequency as a baseline for comparing the ranking described in Section 2.3 – by distributional similarity between a candidate and the original query.

Table 2 shows the average rank of human solutions found in parallel corpora and the recall of these solutions for the top 300 examples. Since there are no substantial differences between the figures for the newspaper texts and for the interviews, we report the results jointly for 556 translation problems in both selections (lower rank figures are better).

	Recall	Average rank
2-word descriptors		
frequency (baseline)	16.7%	*rank=93.7*
distributional similarity	19.5%	*rank=**44.4***
sim. + semantic filter	14.4%	*rank=**26.7***
1-word descriptors		
frequency (baseline)	48.2%	*rank=42.7*
distributional similarity	52.8%	*rank=**21.6***
sim. + semantic filter	44.1%	*rank=**11.3***

Table 2 Ranking: frequency, similarity and filter

It can be seen from the table that ranking by similarity yields almost a twofold improvement for the average rank figures compared to the baseline. There is also a small improvement in recall, since there are more relevant examples that appear within the top 300 entries.

The semantic filter once again gives an almost twofold improvement in ranking, since it removes many noisy items. The average is now within the top 30 items, which means that there is a high chance that a translation solution will be displayed on the first screen. The price for improved ranking is decline in recall, since it may remove some relevant lexical transformations if they appear to be ontologically too far apart. But the decline is smaller: about 26.2% for 2-word descriptors and 16.5% for 1-word descriptors. The semantic filter is an optional tool, which can be used to great effect on noisy output: its improvement of ranking outweighs the decline in recall.

Note that the distribution of ranks is not normal, so in **Figure 1** we present frequency polygons for rank groups of 30 (which is the number of items that fit on a single screen, i.e., the number of items in the first group (r030) shows solutions that will be displayed on the first screen). The majority of solutions ranked by similarity appear high in the list (in fact, on the first two or three screens).

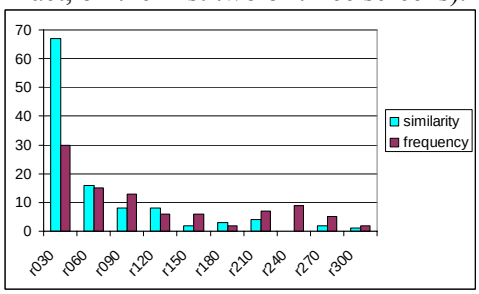


Figure 1 Frequency polygons for ranks

4 Subjective evaluation

The objective evaluation reported above uses a single reference translation and is correspondingly conservative in estimating the coverage of the system. However, many expressions studied have more than one fluent translation. For instance, *in poor repair* is not the only equivalent for the Russian expression *плохо отремонтированные*. It is also possible to translate it as *unsatisfactory condition, bad state of repair, badly in need of repair*, and so on. The objective evaluation shows that the system has been able to find the suggestion used by a particular translator for the problem studied. It does not tell us whether the system has found some other translations suitable for the context. Such legitimate translation variation implies that the performance of a system should be studied on the basis of multiple reference translations, though typically just two reference translations are used (Papineni, et al, 2001). This might be enough for the purposes of a fully automatic MT tool, but in the context of a translator's amanuensis which deals with expressions difficult for human translators, it is reasonable to work with a larger range of acceptable target expressions.

With this in mind we evaluated the performance of the tool with a panel of 12 professional translators. Problematic expressions were highlighted and the translators were asked to find suitable suggestions produced by the tool for these expressions and rank their usability on a scale from 1 to 5 (not acceptable to fully idiomatic, so 1 means that no usable translation was found at all).

Sentences themselves were selected from problems discussed on professional translation forums proz.com and forum.lingvo.ru. Given the range of corpora used in the system (reference and newspa-

per corpora), the examples were filtered to address expressions used in newspapers.

The goal of the subjective evaluation experiment was to establish the usefulness of the system for translators beyond the conservative estimate given by the objective evaluation. The intuition behind the experiment is that if there are several admissible translations for the SL contextual descriptors, and system output matches any of these solutions, then the system has generated something useful. Therefore, we computed recall on sets of human solutions rather than on individual solutions. We matched 210 different human solutions to 36 translation problems. To compute more realistic recall figures, we counted cases when the system output matches any of the human solutions in the set. **Table 3** compares the conservative estimate of the objective evaluation and the more realistic estimate on a single data set.

	2w default	**2w with sem filt**
Conservative	*32.4%; r=53.68*	*21.9%; r=34.67*
Realistic	***75.0%; r=7.48***	***61.1%; r=3.95***

Table 3 Recall and rank for 2-word descriptors

Since the data set is different, the figures for the conservative estimate are higher than those for the objective evaluation data set. However, the table shows the there is a gap between the conservative estimate and the realistic coverage of the translation problems by the system, and that real coverage of indirect translation equivalents is potentially much higher.

Table 4 shows averages (and standard deviation σ) of the usability scores divided in four groups: (1) solutions that are found both by our system and the ORD; (2) solutions found only by our system; (3) solutions found only by ORD (4) solutions found by neither:

	system (+)	**system (–)**
ORD (+)	4.03 (0.42)	3.62 (0.89)
ORD (–)	4.25 (0.79)	3.15 (1.15)

Table 4 Human scores and σ for system output

It can be seen from the table that human users find the system most useful for those problems where the solution does not match any of the direct dictionary equivalents, but is generated by the system.

5 Conclusions

We have presented a method of finding indirect translation equivalents in comparable corpora, and integrated it into a system which assists translators in indirect lexical transfer. The method outperforms established methods of extracting indirect translation equivalents from parallel corpora.

We can interpret these results as an indication that our method, rather than learning individual indirect transformations, models the entire family of transformations entailed by indirect lexical transfer. In other words it learns a translation strategy which is based on the distributional similarity of words in a monolingual corpus, and applies this strategy to novel, previously unseen examples.

The coverage of the tool and additional filtering techniques make it useful for professional translators in automating the search for non-trivial, indirect translation equivalents, especially equivalents for multiword expressions.

References

Gregory Grefenstette. 2002. Multilingual corpus-based extraction and the very large lexicon. In: Lars Borin, editor, Language and Computers, *Parallel corpora, parallel worlds*, pages 137-149. Rodopi.

Martin Kay. 1997. The proper place of men and machines in language translation. *Machine Translation*, 12(1-2):3-23.

Philippe Langlais, Michel Simard, and Jean Véronis. 1998. Methods and practical issues in evaluating alignment techniques. In *Proc. Joint COLING-ACL-98*, pages 711-717.

Jeremy Munday. 2001. *Introducing translation studies. Theories and Applications*. Routledge, New York.

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

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2001). Bleu: a method for automatic evaluation of machine translation, *RC22176 W0109-022*: IBM.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated English and German corpora. In *Procs. the 37th ACL,* pages 395-398.

Reinhard Rapp. 2004. A freely available automatically generated thesaurus of related words. In *Procs. LREC 2004*, pages 395-398, Lisbon.

Serge Sharoff, Bogdan Babych and Anthony Hartley 2006. Using Comparable Corpora to Solve Problems Difficult for Human Translators. In: *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pp. 739-746.

Forest Rescoring: Faster Decoding with Integrated Language Models *

Liang Huang
University of Pennsylvania
Philadelphia, PA 19104
lhuang3@cis.upenn.edu

David Chiang
USC Information Sciences Institute
Marina del Rey, CA 90292
chiang@isi.edu

Abstract

Efficient decoding has been a fundamental problem in machine translation, especially with an integrated language model which is essential for achieving good translation quality. We develop faster approaches for this problem based on k-best parsing algorithms and demonstrate their effectiveness on both phrase-based and syntax-based MT systems. In both cases, our methods achieve significant speed improvements, often by more than a factor of ten, over the conventional beam-search method at the same levels of search error and translation accuracy.

1 Introduction

Recent efforts in statistical machine translation (MT) have seen promising improvements in output quality, especially the phrase-based models (Och and Ney, 2004) and syntax-based models (Chiang, 2005; Galley et al., 2006). However, efficient decoding under these paradigms, especially with integrated language models (LMs), remains a difficult problem. Part of the complexity arises from the expressive power of the translation model: for example, a phrase- or word-based model with full reordering has exponential complexity (Knight, 1999). The language model also, if fully integrated into the decoder, introduces an expensive overhead for maintaining target-language boundary words for dynamic programming (Wu, 1996; Och and Ney, 2004). In practice, one must prune the search space aggressively to reduce it to a reasonable size.

A much simpler alternative method to incorporate the LM is *rescoring*: we first decode without the LM (henceforth *−LM decoding*) to produce a k-best list of candidate translations, and then rerank the k-best list using the LM. This method runs much faster in practice but often produces a considerable number of search errors since the true best translation (taking LM into account) is often outside of the k-best list.

Cube pruning (Chiang, 2007) is a compromise between rescoring and full-integration: it rescores k subtranslations at each node of the forest, rather than only at the root node as in pure rescoring. By adapting the k-best parsing Algorithm 2 of Huang and Chiang (2005), it achieves significant speed-up over full-integration on Chiang's Hiero system.

We push the idea behind this method further and make the following contributions in this paper:

- We generalize cube pruning and adapt it to two systems very different from Hiero: a phrase-based system similar to Pharaoh (Koehn, 2004) and a tree-to-string system (Huang et al., 2006).
- We also devise a faster variant of cube pruning, called *cube growing*, which uses a lazy version of k-best parsing (Huang and Chiang, 2005) that tries to reduce k to the minimum needed at each node to obtain the desired number of hypotheses at the root.

Cube pruning and cube growing are collectively called *forest rescoring* since they both approximately rescore the packed forest of derivations from −LM decoding. In practice they run an order of

* The authors would like to thank Dan Gildea, Jonathan Graehl, Mark Johnson, Kevin Knight, Daniel Marcu, Bob Moore and Hao Zhang. L. H. was partially supported by NSF ITR grants IIS-0428020 while visiting USC/ISI and EIA-0205456 at UPenn. D. C. was partially supported under the GALE/DARPA program, contract HR0011-06-C-0022.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 144–151,
Prague, Czech Republic, June 2007.

magnitude faster than full-integration with beam search, at the same level of search errors and translation accuracy as measured by BLEU.

2 Preliminaries

We establish in this section a unified framework for translation with an integrated n-gram language model in both phrase-based systems and syntax-based systems based on synchronous context-free grammars (SCFGs). An SCFG (Lewis and Stearns, 1968) is a context-free rewriting system for generating string pairs. Each rule $A \rightarrow \alpha, \beta$ rewrites a pair of nonterminals in both languages, where α and β are the source and target side components, and there is a one-to-one correspondence between the nonterminal occurrences in α and the nonterminal occurrences in β. For example, the following rule

$$\text{VP} \rightarrow \text{PP}\ ^{(1)}\ \text{VP}\ ^{(2)}, \quad \text{VP}\ ^{(2)}\ \text{PP}\ ^{(1)}$$

captures the swapping of VP and PP between Chinese (source) and English (target).

2.1 Translation as Deduction

We will use the following example from Chinese to English for both systems described in this section:

yǔ	*Shālóng*	*jǔxíng*	*le*	*huìtán*
with	Sharon	hold	[*past*]	meeting

'held a meeting with Sharon'

A typical phrase-based decoder generates partial target-language outputs in left-to-right order in the form of *hypotheses* (Koehn, 2004). Each hypothesis has a *coverage vector* capturing the source-language words translated so far, and can be extended into a longer hypothesis by a phrase-pair translating an uncovered segment.

This process can be formalized as a deductive system. For example, the following deduction step grows a hypothesis by the phrase-pair ⟨*yǔ Shālóng*, with Sharon⟩:

$$\frac{(\text{\_\_}\bullet\bullet\bullet) : (w, \text{“held a talk”})}{(\bullet\bullet\bullet\bullet\bullet) : (w + c, \text{“held a talk with Sharon”})} \quad (1)$$

where a • in the coverage vector indicates the source word at this position is "covered" (for simplicity we omit here the ending position of the last phrase which is needed for distortion costs), and where w and $w + c$ are the weights of the two hypotheses, respectively, with c being the cost of the phrase-pair.

Similarly, the decoding problem with SCFGs can also be cast as a deductive (parsing) system (Shieber et al., 1995). Basically, we parse the input string using the source projection of the SCFG while building the corresponding subtranslations in parallel. A possible deduction of the above example is notated:

$$\frac{(\text{PP}_{1,3}) : (w_1, t_1) \qquad (\text{VP}_{3,6}) : (w_2, t_2)}{(\text{VP}_{1,6}) : (w_1 + w_2 + c', t_2 t_1)} \quad (2)$$

where the subscripts denote indices in the input sentence just as in CKY parsing, w_1, w_2 are the scores of the two antecedent items, and t_1 and t_2 are the corresponding subtranslations. The resulting translation $t_2 t_1$ is the inverted concatenation as specified by the target-side of the SCFG rule with the additional cost c' being the cost of this rule.

These two deductive systems represent the search space of decoding without a language model. When one is instantiated for a particular input string, it defines a set of derivations, called a *forest*, represented in a compact structure that has a structure of a graph in the phrase-based case, or more generally, a *hypergraph* in both cases. Accordingly we call items like $(\bullet\bullet\bullet\bullet\bullet)$ and $(\text{VP}_{1,6})$ *nodes* in the forest, and instantiated deductions like

$$\begin{aligned} (\bullet\bullet\bullet\bullet\bullet) \quad &\rightarrow \quad (\text{\_\_}\bullet\bullet\bullet) \text{ with Sharon}, \\ (\text{VP}_{1,6}) \quad &\rightarrow \quad (\text{VP}_{3,6})\ (\text{PP}_{1,3}) \end{aligned}$$

we call *hyperedges* that connect one or more antecedent nodes to a consequent node.

2.2 Adding a Language Model

To integrate with a bigram language model, we can use the dynamic-programming algorithms of Och and Ney (2004) and Wu (1996) for phrase-based and SCFG-based systems, respectively, which we may think of as doing a finer-grained version of the deductions above. Each node v in the forest will be split into a set of augmented items, which we call *+LM items*. For phrase-based decoding, a +LM item has the form $(v\ ^{a})$ where a is the last word of the hypothesis. Thus a +LM version of Deduction (1) might be:

$$\frac{(\text{\_\_}\bullet\bullet\bullet\ ^{\text{talk}}) : (w, \text{“held a talk”})}{(\bullet\bullet\bullet\bullet\bullet\ ^{\text{Sharon}}) : (w', \text{“held a talk with Sharon”})}$$

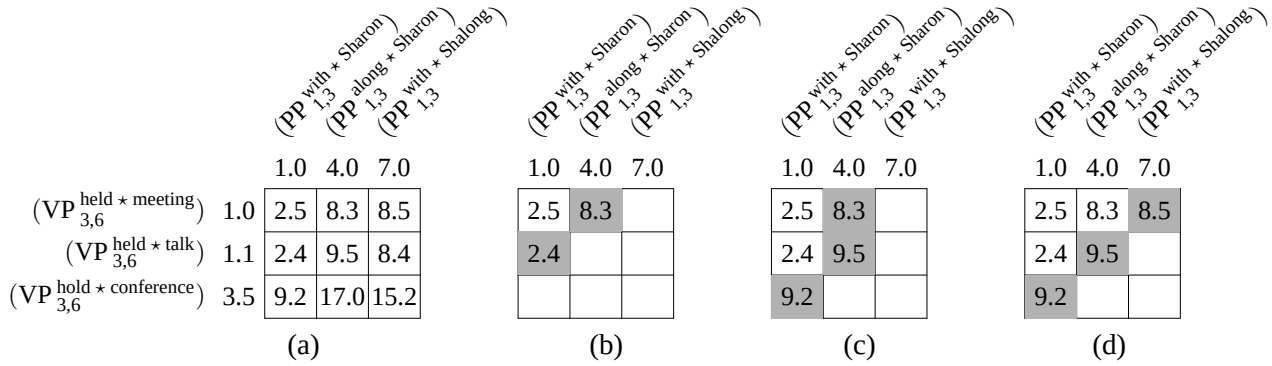


Figure 1: Cube pruning along one hyperedge. (a): the numbers in the grid denote the score of the resulting +LM item, including the combination cost; (b)-(d): the best-first enumeration of the top three items. Notice that the items popped in (b) and (c) are out of order due to the non-monotonicity of the combination cost.

where the score of the resulting +LM item

$$w' = w + c - \log P_{lm}(\text{with} \mid \text{talk})$$

now includes a *combination cost* due to the bigrams formed when applying the phrase-pair.

Similarly, a +LM item in SCFG-based models has the form $(v^{a \star b})$, where a and b are *boundary words* of the hypothesis string, and $\star$ is a placeholder symbol for an elided part of that string, indicating that a possible translation of the part of the input spanned by v starts with a and ends with b. An example +LM version of Deduction (2) is:

$$\frac{(\text{PP}_{1,3}^{\text{with} \star \text{Sharon}})\text{: } (w_1, t_1) \quad (\text{VP}_{3,6}^{\text{held} \star \text{talk}})\text{: } (w_2, t_2)}{(\text{VP}_{1,6}^{\text{held} \star \text{Sharon}})\text{: } (w, t_2 t_1)}$$

where $w = w_1 + w_2 + c' - \log P_{lm}(\text{with} \mid \text{talk})$ with a similar combination cost formed in combining adjacent boundary words of antecedents. This scheme can be easily extended to work with a general n-gram model (Chiang, 2007). The experiments in this paper use trigram models.

The conventional full-integration approach traverses the forest bottom-up and explores all possible +LM deductions along each hyperedge. The theoretical running time of this algorithm is $\mathcal{O}(|F||T|^{(m-1)})$ for phrase-based models, and $\mathcal{O}(|F||T|^{4(m-1)})$ for binary-branching SCFG-based models, where $|F|$ is the size of the forest, and $|T|$ is the number of possible target-side words. Even if we assume a constant number of translations for each word in the input, with a trigram model, this still amounts to $\mathcal{O}(n^{11})$ for SCFG-based models and $\mathcal{O}(2^n n^2)$ for phrase-based models.

3 Cube Pruning

Cube pruning (Chiang, 2007) reduces the search space significantly based on the observation that when the above method is combined with beam search, only a small fraction of the possible +LM items at a node will escape being pruned, and moreover we can select with reasonable accuracy those top-k items without computing all possible items first. In a nutshell, cube pruning works on the −LM forest, keeping at most k +LM items at each node, and uses the k-best parsing Algorithm 2 of Huang and Chiang (2005) to speed up the computation. For simplicity of presentation, we will use concrete SCFG-based examples, but the method applies to the general hypergraph framework in Section 2.

Consider Figure 1(a). Here $k = 3$ and we use $\mathbf{D}(v)$ to denote the top-k +LM items (in sorted order) of node v. Suppose we have computed $\mathbf{D}(u_1)$ and $\mathbf{D}(u_2)$ for the two antecedent nodes $u_1 = (\text{VP}_{3,6})$ and $u_2 = (\text{PP}_{1,3})$ respectively. Then for the consequent node $v = (\text{VP}_{1,6})$ we just need to derive the top-3 from the 9 combinations of $(D_i(u_1), D_j(u_2))$ with $i, j \in [1, 3]$. Since the antecedent items are sorted, it is very likely that the best consequent items in this grid lie towards the upper-left corner. This situation is very similar to k-best parsing and we can adapt the Algorithm 2 of Huang and Chiang (2005) here to explore this grid in a best-first order.

Suppose that the combination costs are negligible, and therefore the weight of a consequent item is just the product of the weights of the antecedent items.

```
1: function CUBE(F)                          ▷ the input is a forest F
2:     for v ∈ F in (bottom-up) topological order do
3:         KBEST(v)
4:     return D_1(TOP)
5: procedure KBEST(v)
6:     cand ← {⟨e, 1⟩ | e ∈ IN(v)}    ▷ for each incoming e
7:     HEAPIFY(cand)                  ▷ a priority queue of candidates
8:     buf ← ∅
9:     while |cand| > 0 and |buf| < k do
10:        item ← POP-MIN(cand)
11:        append item to buf
12:        PUSHSUCC(item, cand)
13:    sort buf to D(v)
14: procedure PUSHSUCC(⟨e, j⟩, cand)
15:    e is v → u_1 ... u_|e|
16:    for i in 1 ... |e| do
17:        j' ← j + b^i
18:        if |D(u_i)| ≥ j'_i then
19:            PUSH(⟨e, j'⟩, cand)
```

Figure 2: Pseudocode for cube pruning.

Then we know that $D_1(v) = (D_1(u_1), D_1(u_2))$, the upper-left corner of the grid. Moreover, we know that $D_2(v)$ is the better of $(D_1(u_1), D_2(u_2))$ and $(D_2(u_1), D_1(u_2))$, the two neighbors of the upper-left corner. We continue in this way (see Figure 1(b)–(d)), enumerating the consequent items best-first while keeping track of a relatively small number of candidates (shaded cells in Figure 1(b), $cand$ in Figure 2) for the next-best item.

However, when we take into account the combination costs, this grid is no longer monotonic in general, and the above algorithm will not always enumerate items in best-first order. We can see this in the first iteration in Figure 1(b), where an item with score 2.5 has been enumerated even though there is an item with score 2.4 still to come. Thus we risk making more search errors than the full-integration method, but in practice the loss is much less significant than the speedup. Because of this disordering, we do not put the enumerated items directly into $\mathbf{D}(v)$; instead, we collect items in a buffer (buf in Figure 2) and re-sort the buffer into $\mathbf{D}(v)$ after it has accumulated k items.[1]

In general the grammar may have multiple rules that share the same source side but have different target sides, which we have treated here as separate hyperedges in the $-$LM forest. In Hiero, these hyperedges are processed as a single unit which we call a *hyperedge bundle*. The different target sides then constitute a third dimension of the grid, forming a cube of possible combinations (Chiang, 2007).

[1] Notice that different combinations might have the same resulting item, in which case we only keep the one with the better score (sometimes called *hypothesis recombination* in MT literature), so the number of items in $\mathbf{D}(v)$ might be less than k.

method	k-best	+LM rescoring...
rescoring	Alg. 3	only at the root node
cube pruning	Alg. 2	on-the-fly at each node
cube growing	Alg. 3	on-the-fly at each node

Table 1: Comparison of the three methods.

Now consider that there are many hyperedges that derive v, and we are only interested the top $+$LM items of v over all incoming hyperedges. Following Algorithm 2, we initialize the priority queue $cand$ with the upper-left corner item from each hyperedge, and proceed as above. See Figure 2 for the pseudocode for cube pruning. We use the notation $\langle e, \mathbf{j} \rangle$ to identify the derivation of v via the hyperedge e and the j_ith best subderivation of antecedent u_i ($1 \leq i \leq |\mathbf{j}|$). Also, we let $\mathbf{1}$ stand for a vector whose elements are all 1, and $\mathbf{b}^i$ for the vector whose members are all 0 except for the ith whose value is 1 (the dimensionality of either should be evident from the context). The heart of the algorithm is lines 10–12. Lines 10–11 move the best derivation $\langle e, \mathbf{j} \rangle$ from $cand$ to buf, and then line 12 pushes its successors $\{\langle e, \mathbf{j} + \mathbf{b}^i \rangle \mid i \in 1 \ldots |e|\}$ into $cand$.

4 Cube Growing

Although much faster than full-integration, cube pruning still computes a fixed amount of $+$LM items at each node, many of which will not be useful for arriving at the 1-best hypothesis at the root. It would be more efficient to compute as few $+$LM items at each node as are needed to obtain the 1-best hypothesis at the root. This new method, called *cube growing*, is a lazy version of cube pruning just as Algorithm 3 of Huang and Chiang (2005), is a lazy version of Algorithm 2 (see Table 1).

Instead of traversing the forest bottom-up, cube growing visits nodes recursively in depth-first order from the root node (Figure 4). First we call LAZYJTHBEST(TOP, 1), which uses the same algorithm as cube pruning to find the 1-best $+$LM item of the root node using the best $+$LM items of

	1.0	4.0	7.0
1.0	2.1	5.1	8.1
1.1	2.2	5.2	8.2
3.5	4.6	7.6	10.6

(a) h-values

1.0	4.0	7.0
2.5	8.3	
2.4		

(b) true costs

Figure 3: Example of cube growing along one hyperedge. (a): the $h(x)$ scores for the grid in Figure 1(a), assuming $h_{combo}(e) = 0.1$ for this hyperedge; (b) cube growing prevents early ranking of the top-left cell (2.5) as the best item in this grid.

the antecedent nodes. However, in this case the best +LM items of the antecedent nodes are not known, because we have not visited them yet. So we recursively invoke LAZYJTHBEST on the antecedent nodes to obtain them as needed. Each invocation of LAZYJTHBEST(v, j) will recursively call itself on the antecedents of v until it is confident that the jth best +LM item for node v has been found.

Consider again the case of one hyperedge e. Because of the nonmonotonicity caused by combination costs, the first +LM item ($\langle e, \mathbf{1} \rangle$) popped from $cand$ is not guaranteed to be the best of all combinations along this hyperedge (for example, the top-left cell of 2.5 in Figure 1 is not the best in the grid). So we cannot simply enumerate items just as they come off of $cand$.[2] Instead, we need to store up popped items in a buffer buf, just as in cube pruning, and enumerate an item only when we are confident that it will never be surpassed in the future. In other words, we would like to have an estimate of the best item not explored yet (analogous to the heuristic function in A* search). If we can establish a lower bound $h_{combo}(e)$ on the combination cost of any +LM deduction via hyperedge e, then we can form a monotonic grid (see Figure 3(a)) of lower bounds on the grid of combinations, by using $h_{combo}(e)$ in place of the true combination cost for each +LM item x in the grid; call this lower bound $h(x)$.

Now suppose that the gray-shaded cells in Figure 3(a) are the members of $cand$. Then the minimum of $h(x)$ over the items in $cand$, in this example, $\min\{2.2, 5.1\} = 2.2$ is a lower bound on the cost of any item in the future for the hyperedge e. Indeed, if $cand$ contains items from multiple hyperedges for a single consequent node, this is still a valid lower bound. More formally:

[2] If we did, then the out-of-order enumeration of +LM items at an antecedent node would cause an entire row or column in the grid to be disordered at the consequent node, potentially leading to a multiplication of search errors.

```
1: procedure LAZYJTHBEST(v, j)
2:     if cand[v] is undefined then
3:         cand[v] ← ∅
4:         FIRE(e, 1, cand) foreach e ∈ IN(v)
5:         buf[v] ← ∅
6:     while |D(v)| < j and |buf[v]| + |D(v)| < k and
       |cand[v]| > 0 do
7:         item ← POP-MIN(cand[v])
8:         PUSH(item, buf[v])
9:         PUSHSUCC(item, cand[v])
10:        bound ← min{h(x) | x ∈ cand[v]}
11:        ENUM(buf[v], D(v), bound)
12:    ENUM(buf[v], D(v), +∞)
13: procedure FIRE(e, j, cand)
14:    e is v → u_1 ... u_|e|
15:    for i in 1 ... |e| do
16:        LAZYJTHBEST(u_i, j_i)
17:        if |D(u_i)| < j_i then return
18:    PUSH(⟨e, j⟩, cand)
19: procedure PUSHSUCC(⟨e, j⟩, cand)
20:    FIRE(e, j + b^i, cand) foreach i in 1 ... |e|
21: procedure ENUM(buf, D, bound)
22:    while |buf| > 0 and MIN(buf) < bound do
23:        append POP-MIN(buf) to D
```

Figure 4: Pseudocode of cube growing.

Lemma 1. *For each node v in the forest, the term*

$$bound = \min_{x \in cand[v]} h(x) \tag{3}$$

is a lower bound on the true cost of any future item that is yet to be explored for v.

Proof. For any item x that is not explored yet, the true cost $c(x) \geq h(x)$, by the definition of h. And there exists an item $y \in cand[v]$ along the same hyperedge such that $h(x) \geq h(y)$, due to the monotonicity of h within the grid along one hyperedge. We also have $h(y) \geq bound$ by the definition of $bound$. Therefore $c(x) \geq bound$. □

Now we can safely pop the best item from buf if its true cost MIN(buf) is better than $bound$ and pass it up to the consequent node (lines 21–23); but otherwise, we have to wait for more items to accumulate in buf to prevent a potential search error, for example, in the case of Figure 3(b), where the top-left cell

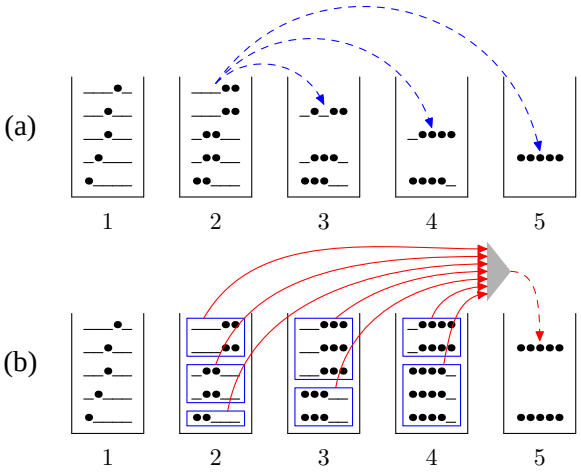


Figure 5: (a) Pharaoh expands the hypotheses in the current bin (#2) into longer ones. (b) In Cubit, hypotheses in previous bins are fed via hyperedge bundles (solid arrows) into a priority queue (shaded triangle), which empties into the current bin (#5).

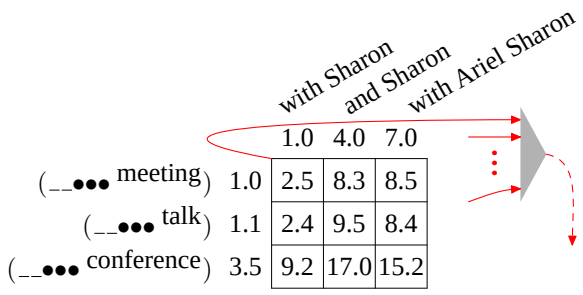


Figure 6: A hyperedge bundle represents all +LM deductions that derives an item in the current bin from the same coverage vector (see Figure 5). The phrases on the top denote the target-sides of applicable phrase-pairs sharing the same source-side.

(2.5) is worse than the current $bound$ of 2.2. The update of $bound$ in each iteration (line 10) can be efficiently implemented by using another heap with the same contents as $cand$ but prioritized by h instead. In practice this is a negligible overhead on top of cube pruning.

We now turn to the problem of estimating the heuristic function h_{combo}. In practice, computing true lower bounds of the combination costs is too slow and would compromise the speed up gained from cube growing. So we instead use a much simpler method that just calculates the minimum combination cost of each hyperedge in the top-i derivations of the root node in $-$LM decoding. This is just an approximation of the true lower bound, and bad estimates can lead to search errors. However, the hope is that by choosing the right value of i, these estimates will be accurate enough to affect the search quality only slightly, which is analogous to "almost admissible" heuristics in A* search (Soricut, 2006).

5 Experiments

We test our methods on two large-scale English-to-Chinese translation systems: a phrase-based system and our tree-to-string system (Huang et al., 2006).

5.1 Phrase-based Decoding

We implemented *Cubit*, a Python clone of the Pharaoh decoder (Koehn, 2004),[3] and adapted cube pruning to it as follows. As in Pharaoh, each bin i contains hypotheses (i.e., +LM items) covering i words on the source-side. But at each bin (see Figure 5), all +LM items from previous bins are first partitioned into $-$LM items; then the hyperedges leading from those $-$LM items are further grouped into hyperedge bundles (Figure 6), which are placed into the priority queue of the current bin.

Our data preparation follows Huang et al. (2006): the training data is a parallel corpus of 28.3M words on the English side, and a trigram language model is trained on the Chinese side. We use the same test set as (Huang et al., 2006), which is a 140-sentence subset of the NIST 2003 test set with 9–36 words on the English side. The weights for the log-linear model are tuned on a separate development set. We set the decoder phrase-table limit to 100 as suggested in (Koehn, 2004) and the distortion limit to 4.

Figure 7(a) compares cube pruning against full-integration in terms of search quality vs. search efficiency, under various pruning settings (threshold beam set to 0.0001, stack size varying from 1 to 200). Search quality is measured by average model cost per sentence (lower is better), and search efficiency is measured by the average number of hypotheses generated (smaller is faster). At each level

[3] In our tests, Cubit always obtains a BLEU score within 0.004 of Pharaoh's (Figure 7(b)). Source code available at `http://www.cis.upenn.edu/~lhuang3/cubit/`

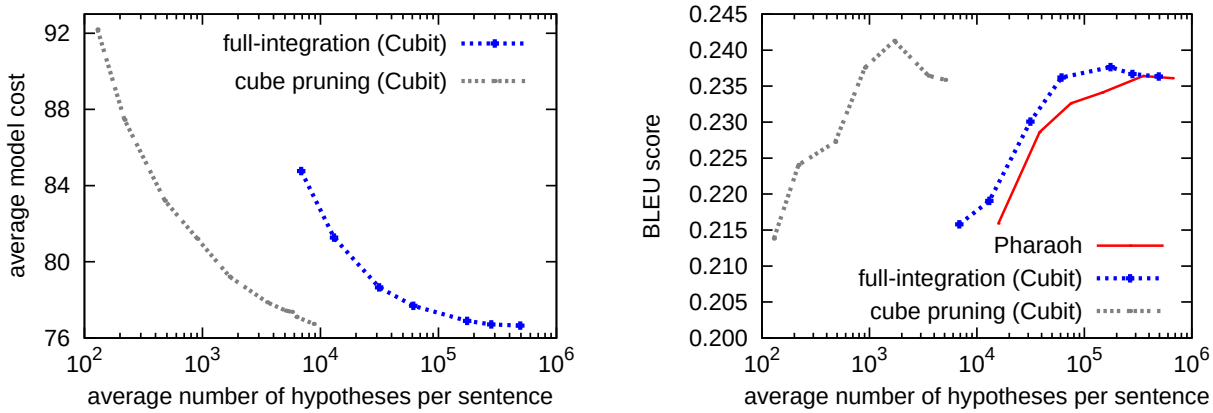
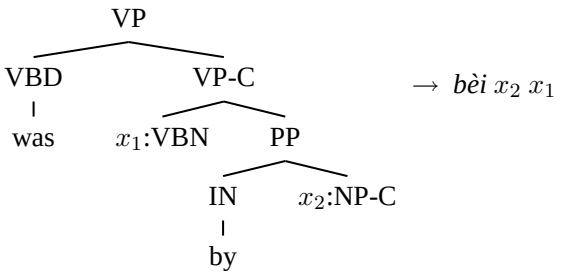


Figure 7: Cube pruning vs. full-integration (with beam search) on phrase-based decoding.

of search quality, the speed-up is always better than a factor of 10. The speed-up at the lowest search-error level is a factor of 32. Figure 7(b) makes a similar comparison but measures search quality by BLEU, which shows an even larger relative speed-up for a given BLEU score, because translations with very different model costs might have similar BLEU scores. It also shows that our full-integration implementation in Cubit faithfully reproduces Pharaoh's performance. Fixing the stack size to 100 and varying the threshold yielded a similar result.

5.2 Tree-to-string Decoding

In tree-to-string (also called *syntax-directed*) decoding (Huang et al., 2006; Liu et al., 2006), the source string is first parsed into a tree, which is then recursively converted into a target string according to transfer rules in a synchronous grammar (Galley et al., 2006). For instance, the following rule translates an English passive construction into Chinese:

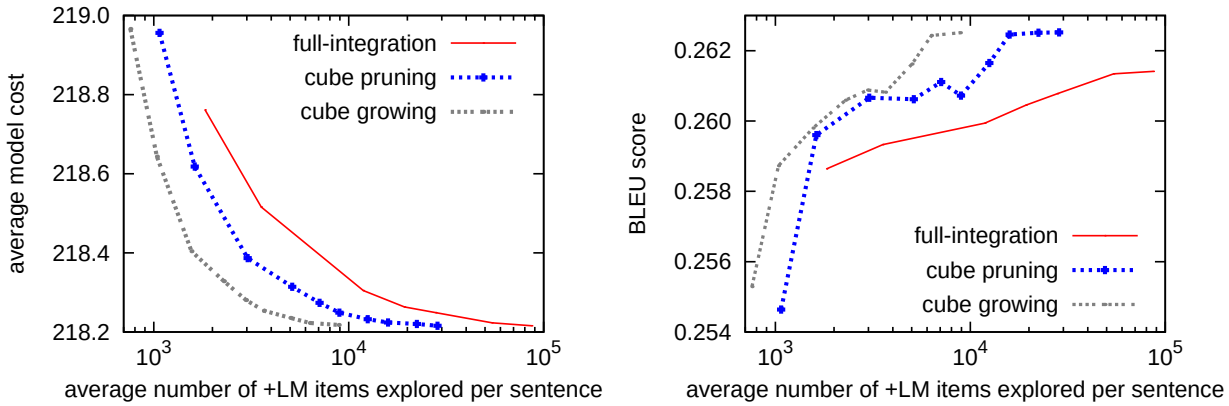


Our tree-to-string system performs slightly better than the state-of-the-art phrase-based system Pharaoh on the above data set. Although different from the SCFG-based systems in Section 2, its derivation trees remain context-free and the search space is still a hypergraph, where we can adapt the methods presented in Sections 3 and 4.

The data set is same as in Section 5.1, except that we also parsed the English-side using a variant of the Collins (1997) parser, and then extracted 24.7M tree-to-string rules using the algorithm of (Galley et al., 2006). Since our tree-to-string rules may have many variables, we first binarize each hyperedge in the forest on the target projection (Huang, 2007). All the three +LM decoding methods to be compared below take these binarized forests as input. For cube growing, we use a non-duplicate k-best method (Huang et al., 2006) to get 100-best unique translations according to −LM to estimate the lower-bound heuristics.[4] This preprocessing step takes on average 0.12 seconds per sentence, which is negligible in comparison to the +LM decoding time.

Figure 8(a) compares cube growing and cube pruning against full-integration under various beam settings in the same fashion of Figure 7(a). At the lowest level of search error, the relative speed-up from cube growing and cube pruning compared with full-integration is by a factor of 9.8 and 4.1, respectively. Figure 8(b) is a similar comparison in terms of BLEU scores and shows an even bigger advantage of cube growing and cube pruning over the baseline.

[4] If a hyperedge is not represented at all in the 100-best −LM derivations at the root node, we use the 1-best −LM derivation of this hyperedge instead. Here, rules that share the same source side but have different target sides are treated as separate hyperedges, not collected into hyperedge bundles, since grouping becomes difficult after binarization.



(a) (b)

Figure 8: Cube growing vs. cube pruning vs. full-integration (with beam search) on tree-to-string decoding.

6 Conclusions and Future Work

We have presented a novel extension of cube pruning called *cube growing*, and shown how both can be seen as general *forest rescoring* techniques applicable to both phrase-based and syntax-based decoding. We evaluated these methods on large-scale translation tasks and observed considerable speed improvements, often by more than a factor of ten. We plan to investigate how to adapt cube growing to phrase-based and hierarchical phrase-based systems.

These forest rescoring algorithms have potential applications to other computationally intensive tasks involving combinations of different models, for example, head-lexicalized parsing (Collins, 1997); joint parsing and semantic role labeling (Sutton and McCallum, 2005); or tagging and parsing with non-local features. Thus we envision forest rescoring as being of general applicability for reducing complicated search spaces, as an alternative to simulated annealing methods (Kirkpatrick et al., 1983).

References

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. ACL*.

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

Michael Collins. 1997. Three generative lexicalised models for statistical parsing. In *Proc. ACL*.

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 *Proc. COLING-ACL*.

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

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proc. AMTA*.

Liang Huang. 2007. Binarization, synchronous binarization, and target-side binarization. In *Proc. NAACL Workshop on Syntax and Structure in Statistical Translation*.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. 1983. Optimization by simulated annealing. *Science*, 220(4598):671–680.

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proc. AMTA*, pages 115–124.

P. M. Lewis and R. E. Stearns. 1968. Syntax-directed transduction. *J. ACM*, 15:465–488.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. COLING-ACL*, pages 609–616.

Franz Joseph Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30:417–449.

Stuart Shieber, Yves Schabes, and Fernando Pereira. 1995. Principles and implementation of deductive parsing. *J. Logic Programming*, 24:3–36.

Radu Soricut. 2006. *Natural Language Generation using an Information-Slim Representation*. Ph.D. thesis, University of Southern California.

Charles Sutton and Andrew McCallum. 2005. Joint parsing and semantic role labeling. In *Proc. CoNLL 2005*.

Dekai Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *Proc. ACL*.

Statistical Machine Translation through Global Lexical Selection and Sentence Reconstruction

Srinivas Bangalore, Patrick Haffner, Stephan Kanthak
AT&T Labs - Research
180 Park Ave, Florham Park, NJ 07932
{srini,haffner,skanthak}@research.att.com

Abstract

Machine translation of a source language sentence involves selecting appropriate target language words and ordering the selected words to form a well-formed target language sentence. Most of the previous work on statistical machine translation relies on (*local*) associations of target words/phrases with source words/phrases for lexical selection. In contrast, in this paper, we present a novel approach to lexical selection where the target words are associated with the entire source sentence (*global*) without the need to compute local associations. Further, we present a technique for reconstructing the target language sentence from the selected words. We compare the results of this approach against those obtained from a finite-state based statistical machine translation system which relies on local lexical associations.

1 Introduction

Machine translation can be viewed as consisting of two subproblems: (a) lexical selection, where appropriate target language lexical items are chosen for each source language lexical item and (b) lexical reordering, where the chosen target language lexical items are rearranged to produce a meaningful target language string. Most of the previous work on statistical machine translation, as exemplified in (Brown et al., 1993), employs word-alignment algorithm (such as GIZA++ (Och and Ney, 2003)) that provides local associations between source and target words. The source-to-target word alignments are sometimes augmented with target-to-source word alignments in order to improve precision. Further, the word-level alignments are extended to phrase-level alignments in order to increase the extent of local associations. The phrasal associations compile some amount of (*local*) lexical reordering of the target words – those permitted by the size of the phrase. Most of the state-of-the-art machine translation systems use phrase-level associations in conjunction with a target language model to produce sentences. There is relatively little emphasis on (*global*) lexical reordering other than the local reorderings permitted within the phrasal alignments. A few exceptions are the hierarchical (possibly syntax-based) transduction models (Wu, 1997; Alshawi et al., 1998; Yamada and Knight, 2001; Chiang, 2005) and the string transduction models (Kanthak et al., 2005).

In this paper, we present an alternate approach to lexical selection and lexical reordering. For lexical selection, in contrast to the local approaches of associating target to source words, we associate target words to the entire source sentence. The intuition is that there may be lexico-syntactic features of the source sentence (not necessarily a single source word) that might trigger the presence of a target word in the target sentence. Furthermore, it might be difficult to exactly associate a target word to a source word in many situations – (a) when the translations are not exact but paraphrases (b) when the target language does not have one lexical item to express the same concept that is expressed by a source word. Extending word to phrase alignments attempts to address some of these situations while alleviating the noise in word-level alignments.

As a consequence of this global lexical selection approach, we no longer have a tight association between source and target language words. The result of lexical selection is simply a bag of words in the target language and the sentence has to be reconstructed using this bag of words. The words in the bag, however, might be enhanced with rich syntactic information that could aid in reconstructing the target sentence. This approach to lexical selection and

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 152–159,
Prague, Czech Republic, June 2007.

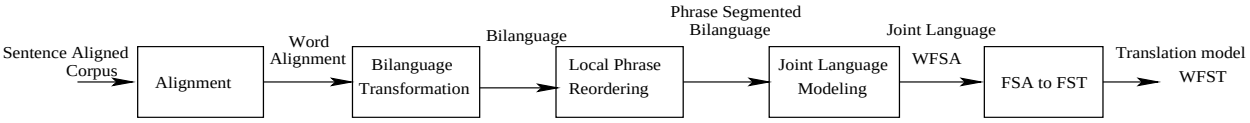


Figure 1: Training phases for our system

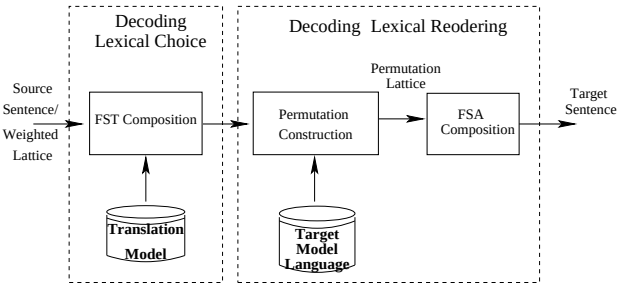


Figure 2: Decoding phases for our system

sentence reconstruction has the potential to circumvent limitations of word-alignment based methods for translation between languages with significantly different word order (e.g. English-Japanese).

In this paper, we present the details of training a global lexical selection model using classification techniques and sentence reconstruction models using permutation automata. We also present a stochastic finite-state transducer (SFST) as an example of an approach that relies on local associations and use it to compare and contrast our approach.

2 SFST Training and Decoding

In this section, we describe each of the components of our SFST system shown in Figure 1. The SFST approach described here is similar to the one described in (Bangalore and Riccardi, 2000) which has subsequently been adopted by (Banchs et al., 2005).

2.1 Word Alignment

The first stage in the process of training a lexical selection model is obtaining an alignment function (f) that given a pair of source ($s_1 s_2 \ldots s_n$) and target ($t_1 t_2 \ldots t_m$) language sentences, maps source language word subsequences into target language word subsequences, as shown below.

$$\forall i \exists j (f(s_i) = t_j \vee f(s_i) = \epsilon) \quad (1)$$

For the work reported in this paper, we have used the GIZA++ tool (Och and Ney, 2003) which implements a string-alignment algorithm. GIZA++ alignment however is asymmetric in that the word mappings are different depending on the direction of alignment – source-to-target or target-to-source. Hence in addition to the functions f as shown in Equation 1 we train another alignment function g :

$$\forall j \exists i (g(t_j) = s_i \vee g(t_j) = \epsilon) \quad (2)$$

English: I need to make a collect call
Japanese: 私は　コレクト　コールを かける　必要があります
Alignment: 1 5 0 3 0 2 4

Figure 3: Example bilingual texts with alignment information

I:私は need:必要があります to:ϵ make:コールを
a:ϵ collect_コレクト call_かける

Figure 4: Bilanguage strings resulting from alignments shown in Figure 3.

2.2 Bilanguage Representation

From the alignment information (see Figure 3), we construct a bilanguage representation of each sentence in the bilingual corpus. The bilanguage string consists of source-target symbol pair sequences as shown in Equation 3. Note that the tokens of a bilanguage could be either ordered according to the word order of the source language or ordered according to the word order of the target language.

$$\begin{aligned} B^f &= b_1^f\, b_2^f\, \ldots\, b_m^f \quad (3) \\ b_i^f &= (s_{i-1}; s_i, f(s_i))\ \ if\ \ f(s_{i-1}) = \epsilon \\ &= (s_i, f(s_{i-1}); f(s_i))\ \ if\ \ s_{i-1} = \epsilon \\ &= (s_i, f(s_i))\ \ otherwise \end{aligned}$$

Figure 4 shows an example alignment and the source-word-ordered bilanguage strings corresponding to the alignment shown in Figure 3.

We also construct a bilanguage using the alignment function g similar to the bilanguage using the alignment function f as shown in Equation 3.

Thus, the bilanguage corpus obtained by combining the two alignment functions is $B = B_f \cup B_g$.

2.3 Bilingual Phrases and Local Reordering

While word-to-word translation only approximates the lexical selection process, phrase-to-phrase mapping can greatly improve the translation of collocations, recurrent strings, etc. Using phrases also allows words within the phrase to be reordered into the correct target language order, thus partially solving the reordering problem. Additionally, SFSTs can take advantage of phrasal correlations to improve the computation of the probability $P(W_S, W_T)$.

The bilanguage representation could result in some source language phrases to be mapped to ϵ

(empty target phrase). In addition to these phrases, we compute subsequences of a given length k on the bilanguage string and for each subsequence we reorder the target words of the subsequence to be in the same order as they are in the target language sentence corresponding to that bilanguage string. This results in a retokenization of the bilanguage into tokens of source-target phrase pairs.

2.4 SFST Model

From the bilanguage corpus B, we train an n-gram language model using standard tools (Goffin et al., 2005). The resulting language model is represented as a weighted finite-state automaton ($S \times T \rightarrow [0,1]$). The symbols on the arcs of this automaton ($s_i\_t_i$) are interpreted as having the source and target symbols (s_i:t_i), making it into a weighted finite-state transducer ($S \rightarrow T \times [0,1]$) that provides a weighted string-to-string transduction from S into T :

$$T^* = \underset{T}{argmax}\ P(s_i, t_i | s_{i-1}, t_{i-1} \ldots s_{i-n-1}, t_{i-n-1})$$

2.5 Decoding

Since we represent the translation model as a weighted finite-state transducer ($TransFST$), the decoding process of translating a new source input (sentence or weighted lattice (I_s)) amounts to a transducer composition ($\circ$) and selection of the best probability path ($BestPath$) resulting from the composition and projecting the target sequence (π_1).

$$T^* = \pi_1(BestPath(I_s \circ TransFST)) \qquad (4)$$

However, we have noticed that on the development corpus, the decoded target sentence is typically shorter than the intended target sentence. This mismatch may be due to the incorrect estimation of the back-off events and their probabilities in the training phase of the transducer. In order to alleviate this mismatch, we introduce a negative word insertion penalty model as a mechanism to produce more words in the target sentence.

2.6 Word Insertion Model

The word insertion model is also encoded as a weighted finite-state automaton and is included in the decoding sequence as shown in Equation 5. The word insertion FST has one state and $|\sum_T|$ number of arcs each weighted with a λ weight representing the word insertion cost. On composition as shown in Equation 5, the word insertion model penalizes or rewards paths which have more words depending on whether λ is positive or negative value.

$$T^* = \pi_1(BestPath(I_s \circ TransFST \circ WIP)) \qquad (5)$$

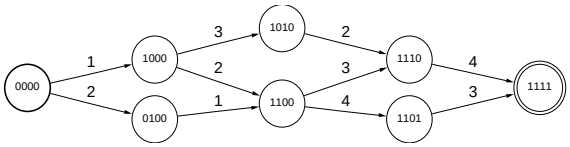


Figure 5: Locally constraint permutation automaton for a sentence with 4 words and window size of 2.

2.7 Global Reordering

Local reordering as described in Section 2.3 is restricted by the window size k and accounts only for different word order within phrases. As permuting non-linear automata is too complex, we apply global reordering by permuting the words of the best translation and weighting the result by an n-gram language model (see also Figure 2):

$$T^* = BestPath(perm(T') \circ LM_t) \qquad (6)$$

Even the size of the minimal permutation automaton of a linear automaton grows exponentially with the length of the input sequence. While decoding by composition simply resembles the principle of memoization (i.e. here: all state hypotheses of a whole sentence are kept in memory), it is necessary to either use heuristic forward pruning or constrain permutations to be within a local window of adjustable size (also see (Kanthak et al., 2005)). We have chosen to constrain permutations here. Figure 5 shows the resulting minimal permutation automaton for an input sequence of 4 words and a window size of 2.

Decoding ASR output in combination with global reordering uses n-best lists or extracts them from lattices first. Each entry of the n-best list is decoded separately and the best target sentence is picked from the union of the n intermediate results.

3 Discriminant Models for Lexical Selection

The approach from the previous section is a generative model for statistical machine translation relying on local associations between source and target sentences. Now, we present our approach for a *global* lexical selection model based on discriminatively trained classification techniques. Discriminant modeling techniques have become the dominant method for resolving ambiguity in speech and other NLP tasks, outperforming generative models. Discriminative training has been used mainly for translation model combination (Och and Ney, 2002) and with the exception of (Wellington et al., 2006; Tillmann and Zhang, 2006), has not been used to directly train parameters of a translation model. We expect discriminatively trained global lexical selection models

to outperform generatively trained local lexical selection models as well as provide a framework for incorporating rich morpho-syntactic information.

Statistical machine translation can be formulated as a search for the best target sequence that maximizes $P(T|S)$, where S is the source sentence and T is the target sentence. Ideally, $P(T|S)$ should be estimated directly to maximize the conditional likelihood on the training data (discriminant model). However, T corresponds to a sequence with a exponentially large combination of possible labels, and traditional classification approaches cannot be used directly. Although Conditional Random Fields (CRF) (Lafferty et al., 2001) train an exponential model at the sequence level, in translation tasks such as ours the computational requirements of training such models are prohibitively expensive.

We investigate two approaches to approximating the string level global classification problem, using different independence assumptions. A comparison of the two approaches is summarized in Table 1.

3.1 Sequential Lexical Choice Model

In the first approach, we formulate a sequential local classification problem as shown in Equations 7. This approach is similar to the SFST approach in that it relies on local associations between the source and target words(phrases). We can use a conditional model (instead of a joint model as before) and the parameters are determined using discriminant training which allows for richer conditioning context.

$$P(T|S) = \prod_{i=1}^{N} P(t_i|\Phi(S,i)) \qquad (7)$$

where $\Phi(S,i)$ is a set of features extracted from the source string S (shortened as Φ in the rest of the section).

3.2 Bag-of-Words Lexical Choice Model

The sequential lexical choice model described in the previous section treats the selection of a lexical choice for a source word in the local lexical context as a classification task. The data for training such models is derived from word alignments obtained by e.g. GIZA++. The decoded target lexical items have to be further reordered, but for closely related languages the reordering could be incorporated into correctly ordered target phrases as discussed previously.

For pairs of languages with radically different word order (e.g. English-Japanese), there needs to be a global reordering of words similar to the case in the SFST-based translation system. Also, for such differing language pairs, the alignment algorithms such as GIZA++ perform poorly.

These observations prompted us to formulate the lexical choice problem *without* the need for word alignment information. We require a sentence aligned corpus as before, but we treat the target sentence as a bag-of-words or BOW assigned to the source sentence. The goal is, given a source sentence, to estimate the probability that we find a given word in the target sentence. This is why, instead of producing a target sentence, what we initially obtain is a target bag of words. Each word in the target vocabulary is detected independently, so we have here a very simple use of binary static classifiers. Training sentence pairs are considered as positive examples when the word appears in the target, and negative otherwise. Thus, the number of training examples equals the number of sentence pairs, in contrast to the sequential lexical choice model which has one training example for each token in the bilingual training corpus. The classifier is trained with n-gram features ($BOgrams(S)$) from the source sentence. During decoding the words with conditional probability greater than a threshold θ are considered as the result of lexical choice decoding.

$$BOW_T^* = \{t|P(t|BOgrams(S)) > \theta\} \qquad (8)$$

For reconstructing the proper order of words in the target sentence we consider all permutations of words in BOW_T^* and weight them by a target language model. This step is similar to the one described in Section 2.7. The BOW approach can also be modified to allow for length adjustments of target sentences, if we add optional deletions in the final step of permutation decoding. The parameter θ and an additional word deletion penalty can then be used to adjust the length of translated outputs. In Section 6, we discuss several issues regarding this model.

4 Choosing the classifier

This section addresses the choice of the classification technique, and argues that one technique that yields excellent performance while scaling well is *binary maximum entropy (Maxent)* with *L1-regularization*.

4.1 Multiclass vs. Binary Classification

The Sequential and BOW models represent two different classification problems. In the sequential model, we have a *multiclass* problem where each class t_i is exclusive, therefore, all the classifier outputs $P(t_i|\Phi)$ must be jointly optimized such that

Table 1: A comparison of the sequential and bag-of-words lexical choice models

	Sequential Lexical Model	**Bag-of-Words Lexical Model**
Output target	Target word for each source position i	Target word given a source sentence
Input features	$BOgram(S, i-d, i+d)$: bag of n-grams in source sentence in the interval $[i-d, i+d]$	$BOgram(S, 0, \|S\|)$: bag of n-grams in source sentence
Probabilities	$P(t_i \| BOgram(S, i-d, i+d))$ Independence assumption between the labels	$P(BOW(T) \| BOgram(S, 0, \|S\|))$
Number of classes	One per target word or phrase	
Training samples	One per source token	One per sentence
Preprocessing	Source/Target *word* alignment	Source/Target *sentence* alignment

$\sum_i P(t_i|\Phi) = 1$. This can be problematic: with one classifier per word in the vocabulary, even allocating the memory during training may exceed the memory capacity of current computers.

In the BOW model, each class can be detected independently, and two different classes can be detected at the same time. This is known as the 1-vs-other scheme. The key advantage over the multiclass scheme is that not all classifiers have to reside in memory at the same time during training which allows for parallelization. Fortunately for the sequential model, we can decompose a multiclass classification problem into separate 1-vs-other problems. In theory, one has to make an additional independence assumption and the problem statement becomes different. Each output label t is projected into a bit string with components $b_j(t)$ where probability of each component is estimated independently:

$$P(b_j(t)|\Phi) = 1 - P(\bar{b}_j(t)|\Phi) = \frac{1}{1 + e^{-(\lambda_j - \lambda_{\bar{j}}) \cdot \Phi}}$$

In practice, despite the approximation, the 1-vs-other scheme has been shown to perform as well as the multiclass scheme (Rifkin and Klautau, 2004). As a consequence, we use the same type of binary classifier for the sequential and the BOW models.

The excellent results recently obtained with the SEARN algorithm (Daume et al., 2007) also suggest that binary classifiers, when properly trained and combined, seem to be capable of matching more complex *structured* output approaches.

4.2 Geometric vs. Probabilistic Interpretation

We separate the most popular classification techniques into two broad categories:

- Geometric approaches maximize the width of a separation margin between the classes. The most popular method is the Support Vector Machine (SVM) (Vapnik, 1998).
- Probabilistic approaches maximize the conditional likelihood of the output class given the input features. This logistic regression is also called Maxent as it finds the distribution with *maximum entropy* that properly estimates the average of each feature over the training data (Berger et al., 1996).

In previous studies, we found that the best accuracy is achieved with non-linear (or kernel) SVMs, at the expense of a high test time complexity, which is unacceptable for machine translation. Linear SVMs and regularized Maxent yield similar performance. In theory, Maxent training, which scales linearly with the number of examples, is faster than SVM training, which scales quadratically with the number of examples. In our first experiments with lexical choice models, we observed that Maxent slightly outperformed SVMs. Using a single threshold with SVMs, some classes of words were over-detected. This suggests that, as theory predicts, SVMs do not properly approximate the posterior probability. We therefore chose to use Maxent as the best probability approximator.

4.3 L1 vs. L2 regularization

Traditionally, Maxent is regularized by imposing a Gaussian prior on each weight: this L2 regularization finds the solution with the smallest possible weights. However, on tasks like machine translation with a very large number of input features, a Laplacian L1 regularization that also attempts to maximize the number of zero weights is highly desirable.

A new L1-regularized Maxent algorithms was proposed for density estimation (Dudik et al., 2004) and we adapted it to classification. We found this algorithm to converge faster than the current state-of-the-art in Maxent training, which is L2-regularized L-BFGS (Malouf, 2002)[1]. Moreover, the number of trained parameters is considerably smaller.

5 Data and Experiments

We have performed experiments on the IWSLT06 Chinese-English training and development sets from

[1]We used the implementation available at http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html

Table 2: Statistics of training and development data from 2005/2006 (* = first of multiple translations only).

	Training (2005)		Dev 2005		Dev 2006	
	Chinese	English	Chinese	English	Chinese	English
Sentences	46,311		506		489	
Running Words	351,060	376,615	3,826	3,897	5,214	6,362*
Vocabulary	11,178	11,232	931	898	1,136	1,134*
Singletons	4,348	4,866	600	538	619	574*
OOVs [%]	-	-	0.6	0.3	0.9	1.0
ASR WER [%]	-	-	-	-	25.2	-
Perplexity	-	-	33	-	86	-
# References	-	-	16		7	

2005 and 2006. The data are traveler task expressions such as seeking directions, expressions in restaurants and travel reservations. Table 2 presents some statistics on the data sets. It must be noted that while the 2005 development set matches the training data closely, the 2006 development set has been collected separately and shows slightly different statistics for average sentence length, vocabulary size and out-of-vocabulary words. Also the 2006 development set contains no punctuation marks in Chinese, but the corresponding English translations have punctuation marks. We also evaluated our models on the Chinese speech recognition output and we report results using 1-best with a word error rate of 25.2%.

For the experiments, we tokenized the Chinese sentences into character strings and trained the models discussed in the previous sections. Also, we trained a punctuation prediction model using Maxent framework on the Chinese character strings in order to insert punctuation marks into the 2006 development data set. The resulting character string with punctuation marks is used as input to the translation decoder. For the 2005 development set, punctuation insertion was not needed since the Chinese sentences already had the true punctuation marks.

In Table 3 we present the results of the three different translation models – FST, Sequential Maxent and BOW Maxent. There are a few interesting observations that can be made based on these results. First, on the 2005 development set, the sequential Maxent model outperforms the FST model, even though the two models were trained starting from the same GIZA++ alignment. The difference, however, is due to the fact that Maxent models can cope with increased lexical context[2] and the parameters of the model are discriminatively trained. The more surprising result is that the BOW Maxent model significantly outperforms the sequential Maxent model. The reason is that the sequential Maxent model relies on the word alignment, which, if erroneous, results in incorrect predictions by the sequential Maxent model. The BOW model does not rely on the word-level alignment and can be interpreted as a discriminatively trained model of dictionary lookup for a target word in the context of a source sentence.

[2]We use 6 words to the left and right of a source word for sequential Maxent, but only 2 preceding source and target words for FST approach.

Table 3: Results (mBLEU) scores for the three different models on the transcriptions for development set 2005 and 2006 and ASR 1-best for development set 2006.

	Dev 2005	Dev 2006	
	Text	Text	ASR 1-best
FST	51.8	19.5	16.5
Seq. Maxent	53.5	19.4	16.3
BOW Maxent	59.9	19.3	16.6

As indicated in the data release document, the 2006 development set was collected differently compared to the one from 2005. Due to this mismatch, the performance of the Maxent models are not very different from the FST model, indicating the lack of good generalization across different genres. However, we believe that the Maxent framework allows for incorporation of linguistic features that could potentially help in generalization across genres. For translation of ASR 1-best, we see a systematic degradation of about 3% in mBLEU score compared to translating the transcription.

In order to compensate for the mismatch between the 2005 and 2006 data sets, we computed a 10-fold average mBLEU score by including 90% of the 2006 development set into the training set and using 10% of the 2006 development set for testing, each time. The average mBLEU score across these 10 runs increased to 22.8.

In Figure 6 we show the improvement of mBLEU scores with the increase in permutation window size. We had to limit to a permutation window size of 10 due to memory limitations, even though the curve has not plateaued. We anticipate using pruning techniques we can increase the window size further.

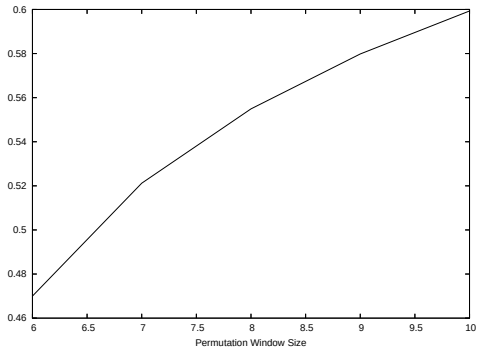


Figure 6: Improvement in mBLEU score with the increase in size of the permutation window

5.1 United Nations and Hansard Corpora

In order to test the scalability of the global lexical selection approach, we also performed lexical selection experiments on the United Nations (Arabic-English) corpus and the Hansard (French-English) corpus using the SFST model and the BOW Maxent model. We used 1,000,000 training sentence pairs and tested on 994 test sentences for the UN corpus. For the Hansard corpus we used the same training and test split as in (Zens and Ney, 2004): 1.4 million training sentence pairs and 5432 test sentences. The vocabulary sizes for the two corpora are mentioned in Table 4. Also in Table 4, are the results in terms of F-measure between the words in the reference sentence and the decoded sentences. We can see that the BOW model outperforms the SFST model on both corpora significantly. This is due to a systematic 10% relative improvement for open class words, as they benefit from a much wider context. BOW performance on close class words is higher for the UN corpus but lower for the Hansard corpus.

Table 4: Lexical Selection results (F-measure) on the Arabic-English UN Corpus and the French-English Hansard Corpus. In parenthesis are F-measures for open and closed class lexical items.

Corpus	Vocabulary		SFST	BOW
	Source	Target		
UN	252,571	53,005	64.6 (60.5/69.1)	69.5 (66.2/72.6)
Hansard	100,270	78,333	57.4 (50.6/67.7)	60.8 (56.5/63.4)

6 Discussion

The BOW approach is promising as it performs reasonably well despite considerable losses in the transfer of information between source and target language. The first and most obvious loss is about word position. The only information we currently use to restore the target word position is the target language model. Information about the grammatical role of a word in the source sentence is completely lost. The language model might fortuitously recover this information if the sentence with the correct grammatical role for the word happens to be the maximum likelihood sentence in the permutation automaton.

We are currently working toward incorporating syntactic information on the target words so as to be able to recover some of the grammatical role information lost in the classification process. In preliminary experiments, we have associated the target lexical items with supertag information (Bangalore and Joshi, 1999). Supertags are labels that provide linear ordering constraints as well as grammatical relation information. Although associating supertags to target words increases the class set for the classifier, we have noticed that the degradation in the F-score is on the order of 3% across different corpora. The supertag information can then be exploited in the sentence construction process. The use of supertags in phrase-based SMT system has been shown to improve results (Hassan et al., 2006).

A less obvious loss is the number of times a word or concept appears in the target sentence. *Function words* like "the" and "of" can appear many times in an English sentence. In the model discussed in this paper, we index each occurrence of the function word with a counter. In order to improve this method, we are currently exploring a technique where the function words serve as attributes (e.g. definiteness, tense, case) on the contentful lexical items, thus enriching the lexical item with morpho-syntactic information.

A third issue concerning the BOW model is the problem of *synonyms* – target words which translate the same source word. Suppose that in the training data, target words t_1 and t_2 are, with equal probability, translations of the same source word. Then, in the presence of this source word, the probability to detect the corresponding target word, which we assume is 0.8, will be, because of discriminant learning, split equally between t_1 and t_2, that is 0.4 and 0.4. Because of this synonym problem, the BOW threshold θ has to be set lower than 0.5, which is observed experimentally. However, if we set the threshold to 0.3, both t_1 and t_2 will be detected in the target sentence, and we found this to be a major source of undesirable insertions.

The BOW approach is different from the parsing based approaches (Melamed, 2004; Zhang and Gildea, 2005; Cowan et al., 2006) where the translation model tightly couples the syntactic and lexical items of the two languages. The decoupling of the

two steps in our model has the potential for generating paraphrased sentences not necessarily isomorphic to the structure of the source sentence.

7 Conclusions

We view machine translation as consisting of lexical selection and lexical reordering steps. These two steps need not necessarily be sequential and could be tightly integrated. We have presented the weighted finite-state transducer model of machine translation where lexical choice and a limited amount of lexical reordering are tightly integrated into a single transduction. We have also presented a novel approach to translation where these two steps are loosely coupled and the parameters of the lexical choice model are discriminatively trained using a maximum entropy model. The lexical reordering model in this approach is achieved using a permutation automaton. We have evaluated these two approaches on the 2005 and 2006 IWSLT development sets and shown that the techniques scale well to Hansard and UN corpora.

References

H. Alshawi, S. Bangalore, and S. Douglas. 1998. Automatic acquisition of hierarchical transduction models for machine translation. In *ACL*, Montreal, Canada.

R.E. Banchs, J.M. Crego, A. Gispert, P. Lambert, and J.B. Marino. 2005. Statistical machine translation of euparl data by using bilingual n-grams. In *Workshop on Building and Using Parallel Texts*. ACL.

S. Bangalore and A. K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2).

S. Bangalore and G. Riccardi. 2000. Stochastic finite-state models for spoken language machine translation. In *Proceedings of the Workshop on Embedded Machine Translation Systems*, pages 52–59.

A.L. Berger, Stephen A. D. Pietra, D. Pietra, and J. Vincent. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71.

P. Brown, S.D. Pietra, V.D. Pietra, and R. Mercer. 1993. The Mathematics of Machine Translation: Parameter Estimation. *Computational Linguistics*, 16(2):263–312.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the ACL Conference*, Ann Arbor, MI.

B. Cowan, I. Kucerova, and M. Collins. 2006. A discriminative model for tree-to-tree translation. In *Proceedings of EMNLP*.

H. Daume, J. Langford, and D. Marcu. 2007. Search-based structure prediction. *submitted to Machine Learning Journal*.

M. Dudik, S. Phillips, and R.E. Schapire. 2004. Performance Guarantees for Regularized Maximum Entropy Density Estimation. In *Proceedings of COLT'04*, Banff, Canada. Springer Verlag.

V. Goffin, C. Allauzen, E. Bocchieri, D. Hakkani-Tur, A. Ljolje, S. Parthasarathy, M. Rahim, G. Riccardi, and M. Saraclar. 2005. The AT&T WATSON Speech Recognizer. In *Proceedings of ICASSP*, Philadelphia, PA.

H. Hassan, M. Hearne, K. Sima'an, and A. Way. 2006. Syntactic phrase-based statistical machine translation. In *Proceedings of IEEE/ACL first International Workshop on Spoken Language Technology (SLT)*, Aruba, December.

S. Kanthak, D. Vilar, E. Matusov, R. Zens, and H. Ney. 2005. Novel reordering approaches in phrase-based statistical machine translation. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 167–174, Ann Arbor, Michigan.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, San Francisco, CA.

R. Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of CoNLL-2002*, pages 49–55. Taipei, Taiwan.

I. D. Melamed. 2004. Statistical machine translation by parsing. In *Proceedings of ACL*.

F. J. Och and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of ACL*.

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

Ryan Rifkin and Aldebaro Klautau. 2004. In defense of one-vs-all classification. *Journal of Machine Learning Research*, pages 101–141.

C. Tillmann and T. Zhang. 2006. A discriminative global training algorithm for statistical mt. In *COLING-ACL*.

V.N. Vapnik. 1998. *Statistical Learning Theory*. John Wiley & Sons.

B. Wellington, J. Turian, C. Pike, and D. Melamed. 2006. Scalable purely-discriminative training for word and tree transducers. In *AMTA*.

D. Wu. 1997. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, 23(3):377–404.

K. Yamada and K. Knight. 2001. A syntax-based statistical translation model. In *Proceedings of* 39^{th} *ACL*.

R. Zens and H. Ney. 2004. Improvements in phrase-based statistical machine translation. In *Proceedings of HLT-NAACL*, pages 257–264, Boston, MA.

H. Zhang and D. Gildea. 2005. Stochastic lexicalized inversion transduction grammar for alignment. In *Proceedings of ACL*.

Mildly Context-Sensitive Dependency Languages

Marco Kuhlmann
Programming Systems Lab
Saarland University
Saarbrücken, Germany
kuhlmann@ps.uni-sb.de

Mathias Möhl
Programming Systems Lab
Saarland University
Saarbrücken, Germany
mmohl@ps.uni-sb.de

Abstract

Dependency-based representations of natural language syntax require a fine balance between structural flexibility and computational complexity. In previous work, several constraints have been proposed to identify classes of dependency structures that are well-balanced in this sense; the best-known but also most restrictive of these is projectivity. Most constraints are formulated on fully specified structures, which makes them hard to integrate into models where structures are composed from lexical information. In this paper, we show how two empirically relevant relaxations of projectivity can be lexicalized, and how combining the resulting lexicons with a regular means of syntactic composition gives rise to a hierarchy of mildly context-sensitive dependency languages.

1 Introduction

Syntactic representations based on word-to-word dependencies have a long tradition in descriptive linguistics. Lately, they have also been used in many computational tasks, such as relation extraction (Culotta and Sorensen, 2004), parsing (McDonald et al., 2005), and machine translation (Quirk et al., 2005).

Especially in recent work on parsing, there is a particular interest in *non-projective* dependency structures, in which a word and its dependents may be spread out over a discontinuous region of the sentence. These structures naturally arise in the syntactic analysis of languages with flexible word order, such as Czech (Veselá et al., 2004). Unfortunately, most formal results on non-projectivity are discouraging: While grammar-driven dependency parsers that are restricted to projective structures can be as efficient as parsers for lexicalized context-free grammar (Eisner and Satta, 1999), parsing is prohibitively expensive when unrestricted forms of non-projectivity are permitted (Neuhaus and Bröker, 1997). Data-driven dependency parsing with non-projective structures is quadratic when all attachment decisions are assumed to be independent of one another (McDonald et al., 2005), but becomes intractable when this assumption is abandoned (McDonald and Pereira, 2006).

In search of a balance between structural flexibility and computational complexity, several authors have proposed constraints to identify classes of non-projective dependency structures that are computationally well-behaved (Bodirsky et al., 2005; Nivre, 2006). In this paper, we focus on two of these proposals: the *gap-degree restriction*, which puts a bound on the number of discontinuities in the region of a sentence covered by a word and its dependents, and the *well-nestedness condition*, which constrains the arrangement of dependency subtrees. Both constraints have been shown to be in very good fit with data from dependency treebanks (Kuhlmann and Nivre, 2006). However, like all other such proposals, they are formulated on fully specified structures, which makes it hard to integrate them into a generative model, where dependency structures are composed from elementary units of lexicalized information. Consequently, little is known about the generative capacity and computational complexity of *languages* over restricted non-projective dependency structures.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 160–167,
Prague, Czech Republic, June 2007.

Contents of the paper In this paper, we show how the gap-degree restriction and the well-nestedness condition can be captured in dependency lexicons, and how combining such lexicons with a regular means of syntactic composition gives rise to an infinite hierarchy of mildly context-sensitive languages.

The technical key to these results is a procedure to encode arbitrary, even non-projective dependency structures into trees (terms) over a signature of local order-annotations. The constructors of these trees can be read as lexical entries, and both the gap-degree restriction and the well-nestedness condition can be couched as syntactic properties of these entries. Sets of gap-restricted dependency structures can be described using regular tree grammars. This gives rise to a notion of *regular dependency languages*, and allows us to establish a formal relation between the structural constraints and mildly context-sensitive grammar formalisms (Joshi, 1985): We show that regular dependency languages correspond to the sets of derivations of lexicalized Linear Context-Free Rewriting Systems (LCFRS) (Vijay-Shanker et al., 1987), and that the gap-degree measure is the structural correspondent of the concept of 'fan-out' in this formalism (Satta, 1992). We also show that adding the well-nestedness condition corresponds to the restriction of LCFRS to Coupled Context-Free Grammars (Hotz and Pitsch, 1996), and that regular sets of well-nested structures with a gap-degree of at most 1 are exactly the class of sets of derivations of Lexicalized Tree Adjoining Grammar (LTAG). This result generalizes previous work on the relation between LTAG and dependency representations (Rambow and Joshi, 1997; Bodirsky et al., 2005).

Structure of the paper The remainder of this paper is structured as follows. Section 2 contains some basic notions related to trees and dependency structures. In Section 3 we present the encoding of dependency structures as order-annotated trees, and show how this encoding allows us to give a lexicalized reformulation of both the gap-degree restriction and the well-nestedness condition. Section 4 introduces the notion of regular dependency languages. In Section 5 we show how different combinations of restrictions on non-projectivity in these languages correspond to different mildly context-sensitive grammar formalisms. Section 6 concludes the paper.

2 Preliminaries

Throughout the paper, we write $[n]$ for the set of all positive natural numbers up to and including n. The set of all strings over a set A is denoted by A^*, the empty string is denoted by ε, and the concatenation of two strings x and y is denoted either by xy, or, where this is ambiguous, by $x \cdot y$.

2.1 Trees

In this paper, we regard trees as terms. We expect the reader to be familiar with the basic concepts related to this framework, and only introduce our particular notation. Let Σ be a set of labels. The set of (finite, unranked) *trees* over Σ is defined recursively by the equation $T_\Sigma := \{\, \sigma(x) \mid \sigma \in \Sigma, x \in T_\Sigma^* \,\}$. The set of *nodes* of a tree $t \in T_\Sigma$ is defined as

$$N(\sigma(t_1 \cdots t_n)) := \{\varepsilon\} \cup \{\, iu \mid i \in [n], u \in N(t_i) \,\}.$$

For two nodes $u, v \in N(t)$, we say that u *governs* v, and write $u \trianglelefteq v$, if v can be written as $v = ux$, for some sequence $x \in \mathbb{N}^*$. Note that the governance relation is both reflexive and transitive. The converse of government is called *dependency*, so $u \trianglelefteq v$ can also be read as 'v depends on u'. The *yield* of a node $u \in N(t)$, $\lfloor u \rfloor$, is the set of all dependents of u in t: $\lfloor u \rfloor := \{\, v \in N(t) \mid u \trianglelefteq v \,\}$. We also use the notations $t(u)$ for the label at the node u of t, and t/u for the subtree of t rooted at u. A *tree language* over Σ is a subset of T_Σ.

2.2 Dependency structures

For the purposes of this paper, a *dependency structure* over Σ is a pair $d = (t, x)$, where $t \in T_\Sigma$ is a tree, and x is a list of the nodes in t. We write D_Σ to refer to the set of all dependency structures over Σ. Independently of the governance relation in d, the list x defines a total order on the nodes in t; we write $u \preceq v$ to denote that u *precedes* v in this order. Note that, like governance, the precedence relation is both reflexive and transitive. A *dependency language* over Σ is a subset of D_Σ.

EXAMPLE. The left half of Figure 1 shows how we visualize dependency structures: circles represent nodes, arrows represent the relation of (immediate) governance, the left-to-right order of the nodes represents their order in the precedence relation, and the dotted lines indicate the labelling. □

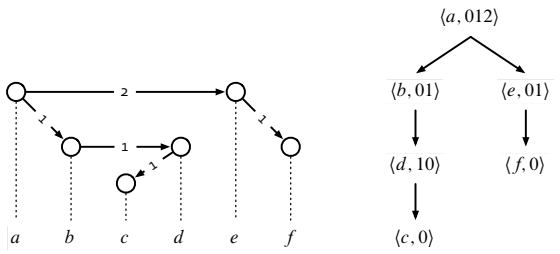


Figure 1: A projective dependency structure

3 Lexicalizing the precedence relation

In this section, we show how the precedence relation of dependency structures can be encoded as, and decoded from, a collection of node-specific order annotations. Under the assumption that the nodes of a dependency structure correspond to lexemic units, this result demonstrates how word-order information can be captured in a dependency lexicon.

3.1 Projective structures

Lexicalizing the precedence relation of a dependency structure is particularly easy if the structure under consideration meets the condition of *projectivity*. A dependency structure is projective, if each of its yields forms an interval with respect to the precedence order (Kuhlmann and Nivre, 2006).

In a projective structure, the interval that corresponds to a yield $\lfloor u \rfloor$ decomposes into the singleton interval $[u, u]$, and the collection of the intervals that correspond to the yields of the immediate dependents of u. To reconstruct the global precedence relation, it suffices to annotate each node u with the relative precedences among the constituent parts of its yield. We represent this 'local' order as a string over the alphabet $\mathbb{N}_0$, where the symbol 0 represents the singleton interval $[u, u]$, and a symbol $i \neq 0$ represents the interval that corresponds to the yield of the ith direct dependent of u. An *order-annotated tree* is a tree labelled with pairs $\langle \sigma, \omega \rangle$, where σ is the label proper, and ω is a local order annotation. In what follows, we will use the functional notations $\sigma(u)$ and $\omega(u)$ to refer to the label and order annotation of u, respectively.

EXAMPLE. Figure 1 shows a projective dependency structure together with its representation as an order-annotated tree. □

We now present procedures for encoding projective dependency structures into order-annotated trees, and for reversing this encoding.

Encoding The representation of a projective dependency structure (t, x) as an order-annotated tree can be computed in a single left-to-right sweep over x. Starting with a copy of the tree t in which every node is annotated with the empty string, for each new node u in x, we update the order annotation of u through the assignment $\omega(u) := \omega(u) \cdot 0$. If $u = vi$ for some $i \in \mathbb{N}$ (that is, if u is an inner node), we also update the order annotation of the parent v of u through the assignment $\omega(v) := \omega(v) \cdot i$.

Decoding To decode an order-annotated tree t, we first linearize the nodes of t into a sequence x, and then remove all order annotations. Linearization proceeds in a way that is very close to a pre-order traversal of the tree, except that the relative position of the root node of a subtree is explicitly specified in the order annotation. Specifically, to linearize an order-annotated tree, we look into the local order $\omega(u)$ annotated at the root node of the tree, and concatenate the linearizations of its constituent parts. A symbol i in $\omega(u)$ represents either the singleton interval $[u, u]$ ($i = 0$), or the interval corresponding to some direct dependent ui of u ($i \neq 0$), in which case we proceed recursively. Formally, the linearization of u is captured by the following three equations:

$$\begin{aligned} \mathrm{lin}(u) &= \mathrm{lin}'(u, \omega(u)) \\ \mathrm{lin}'(u, i_1 \cdots i_n) &= \mathrm{lin}''(u, i_1) \cdots \mathrm{lin}''(u, i_n) \\ \mathrm{lin}''(u, i) &= \textbf{if } i = 0 \textbf{ then } u \textbf{ else } \mathrm{lin}(ui) \end{aligned}$$

Both encoding and decoding can be done in time linear in the number of nodes of the dependency structure or order-annotated tree.

3.2 Non-projective structures

It is straightforward to see that our representation of dependency structures is insufficient if the structures under consideration are non-projective. To witness, consider the structure shown in Figure 2. Encoding this structure using the procedure presented above yields the same order-annotated tree as the one shown in Figure 1, which demonstrates that the encoding is not reversible.

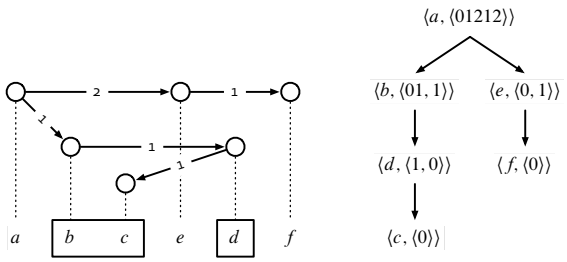


Figure 2: A non-projective dependency structure

Blocks In a non-projective dependency structure, the yield of a node may be spread out over more than one interval; we will refer to these intervals as *blocks*. Two nodes v, w belong to the same block of a node u, if all nodes between v and w are governed by u.

EXAMPLE. Consider the nodes b, c, d in the structures depicted in Figures 1 and 2. In Figure 1, these nodes belong to the same block of b. In Figure 2, the three nodes are spread out over two blocks of b (marked by the boxes): c and d are separated by a node (e) not governed by b. □

Blocks have a recursive structure that is closely related to the recursive structure of yields: the blocks of a node u can be decomposed into the singleton $[u, u]$, and the blocks of the direct dependents of u. Just as a projective dependency structure can be represented by annotating each yield with an order on its constituents, an unrestricted structure can be represented by annotating each block.

Extended order annotations To represent orders on blocks, we extend our annotation scheme as follows. First, instead of a single string, an annotation $\omega(u)$ now is a tuple of strings, where the kth component specifies the order among the constituents of the kth block of u. Second, instead of one, the annotation may now contain multiple occurrences of the same dependent; the kth occurrence of i in $\omega(u)$ represents the kth block of the node ui.

We write $\omega(u)_k$ to refer to the kth component of the order annotation of u. We also use the notation $(i\#k)_u$ to refer to the kth occurrence of i in $\omega(u)$, and omit the subscript when the node u is implicit.

EXAMPLE. In the annotated tree shown in Figure 2, $\omega(b)_1 = (0\#1)(1\#1)$, and $\omega(b)_2 = (1\#2)$. □

Encoding To encode a dependency structure (t, x) as an extended order-annotated tree, we do a post-order traversal of t as follows. For a given node u, let us represent a constituent of a block of u as a triple $i : [v_l, v_r]$, where i denotes the node that contributes the constituent, and v_l and v_r denote the constituent's leftmost and rightmost elements. At each node u, we have access to the singleton block $0 : [u, u]$, and the constituent blocks of the immediate dependents of u. We say that two blocks $i : [v_l, v_r]$, $j : [w_l, w_r]$ can be *merged*, if the node v_r immediately precedes the node w_l. The result of the merger is a new block $ij : [v_l, w_r]$ that represents the information that the two merged constituents belong to the same block of u. By exhaustive merging, we obtain the constituent structure of all blocks of u. From this structure, we can read off the order annotation $\omega(u)$.

EXAMPLE. The yield of the node b in Figure 2 decomposes into $0 : [b, b]$, $1 : [c, c]$, and $1 : [d, d]$. Since b and c are adjacent, the first two of these constituents can be merged into a new block $01 : [b, c]$; the third constituent remains unchanged. This gives rise to the order annotation $\langle 01, 1\rangle$ for b. □

When using a global data-structure to keep track of the constituent blocks, the encoding procedure can be implemented to run in time linear in the number of blocks in the dependency structure. In particular, for projective dependency structures, it still runs in time linear in the number of nodes.

Decoding To linearize the kth block of a node u, we look into the kth component of the order annotated at u, and concatenate the linearizations of its constituent parts. Each occurrence $(i\#k)$ in a component of $\omega(u)$ represents either the node u itself ($i = 0$), or the kth block of some direct dependent ui of u ($i \neq 0$), in which case we proceed recursively:

$$\begin{aligned} \mathrm{lin}(u, k) &= \mathrm{lin}'(u, \omega(u)_k) \\ \mathrm{lin}'(u, i_1 \cdots i_n) &= \mathrm{lin}''(u, i_1) \cdots \mathrm{lin}''(u, i_n) \\ \mathrm{lin}''(u, (i\#k)_u) &= \textbf{if } i = 0 \textbf{ then } u \textbf{ else } \mathrm{lin}(ui, k) \end{aligned}$$

The root node of a dependency structure has only one block. Therefore, to linearize a tree t, we only need to linearize the first block of the tree's root node: $\mathrm{lin}(t) = \mathrm{lin}(\varepsilon, 1)$.

Consistent order annotations Every dependency structure over Σ can be encoded as a tree over the set $\Sigma \times \Omega$, where Ω is the set of all order annotations. The converse of this statement does not hold: to be interpretable as a dependency structure, tree structure and order annotation in an order-annotated tree must be *consistent*, in the following sense.

PROPERTY C1: Every annotation $\omega(u)$ in a tree t contains all and only the symbols in the collection $\{0\} \cup \{\, i \mid ui \in N(t) \,\}$, i.e., one symbol for u, and one symbol for every direct dependent of u.

PROPERTY C2: The number of occurrences of a symbol $i \neq 0$ in $\omega(u)$ is identical to the number of components in the annotation of the node ui. Furthermore, the number of components in the annotation of the root node is 1.

With this notion of consistency, we can prove the following technical result about the relation between dependency structures and annotated trees. We write $\pi_\Sigma(s)$ for the tree obtained from a tree $s \in T_{\Sigma\times\Omega}$ by re-labelling every node u with $\sigma(u)$.

PROPOSITION 1. For every dependency structure (t, x) over Σ, there exists a tree s over $\Sigma \times \Omega$ such that $\pi_\Sigma(s) = t$ and $\mathrm{lin}(s) = x$. Conversely, for every consistently order-annotated tree $s \in T_{\Sigma\times\Omega}$, there exists a uniquely determined dependency structure (t, x) with these properties. □

3.3 Local versions of structural constraints

The encoding of dependency structures as order-annotated trees allows us to reformulate two constraints on non-projectivity originally defined on fully specified dependency structures (Bodirsky et al., 2005) in terms of syntactic properties of the order annotations that they induce:

Gap-degree The *gap-degree* of a dependency structure is the maximum over the number of discontinuities in any yield of that structure.

EXAMPLE. The structure depicted in Figure 2 has gap-degree 1: the yield of b has one discontinuity, marked by the node e, and this is the maximal number of discontinuities in any yield of the structure. □

Since a discontinuity in a yield is delimited by two blocks, and since the number of blocks of a node u equals the number of components in the order annotation of u, the following result is obvious:

PROPOSITION 2. A dependency structure has gap-degree k if and only if the maximal number of components among the annotations $\omega(u)$ is $k + 1$. □

In particular, a dependency structure is projective iff all of its annotations consist of just one component.

Well-nestedness The well-nestedness condition constrains the arrangement of subtrees in a dependency structure. Two subtrees $t/u_1, t/u_2$ *interleave*, if there are nodes $v_l^1, v_r^1 \in t/u_1$ and $v_l^2, v_r^2 \in t/u_2$ such that $v_l^1 \prec v_l^2 \prec v_r^1 \prec v_r^2$. A dependency structure is *well-nested*, if no two of its disjoint subtrees interleave. We can prove the following result:

PROPOSITION 3. A dependency structure is well-nested if and only if no annotation $\omega(u)$ contains a substring $i \cdots j \cdots i \cdots j$, for $i, j \in \mathbb{N}$. □

EXAMPLE. The dependency structure in Figure 1 is well-nested, the structure depicted in Figure 2 is not: the subtrees rooted at the nodes b and e interleave. To see this, notice that $b \prec e \prec d \prec f$. Also notice that $\omega(a)$ contains the substring 1212. □

4 Regular dependency languages

The encoding of dependency structures as order-annotated trees gives rise to an encoding of dependency languages as tree languages. More specifically, dependency languages over a set Σ can be encoded as tree languages over the set $\Sigma \times \Omega$, where Ω is the set of all order annotations. Via this encoding, we can study dependency languages using the tools and results of the well-developed formal theory of tree languages. In this section, we discuss dependency languages that can be encoded as regular tree languages.

4.1 Regular tree grammars

The class of regular tree languages, REGT for short, is a very natural class with many characterizations (Gécseg and Steinby, 1997): it is generated by regular tree grammars, recognized by finite tree automata, and expressible in monadic second-order logic. Here we use the characterization in terms of grammars. Regular tree grammars are natural candidates for the formalization of dependency lexicons, as each rule in such a grammar can be seen as the specification of a word and the syntactic categories or grammatical functions of its immediate dependents.

Formally, a *(normalized) regular tree grammar* is a construct $G = (N_G, \Sigma_G, S_G, P_G)$, in which N_G and Σ_G are finite sets of non-terminal and terminal symbols, respectively, $S_G \in N_G$ is a dedicated start symbol, and P_G is a finite set of productions of the form $A \to \sigma(A_1 \cdots A_n)$, where $\sigma \in \Sigma_G$, $A \in N_G$, and $A_i \in N_G$, for every $i \in [n]$. The *(direct) derivation relation* associated to G is the binary relation $\Rightarrow_G$ on the set $T_{\Sigma_G \cup N_G}$ defined as follows:

$$\frac{t \in T_{\Sigma_G \cup N_G} \qquad t/u = A \qquad (A \to s) \in P_G}{t \Rightarrow_G t[u \mapsto s]}$$

Informally, each step in a derivation replaces a non-terminal-labelled leaf by the right-hand side of a matching production. The *tree language* generated by G is the set of all terminal trees that can eventually be derived from the trivial tree formed by its start symbol: $L(G) = \{\, t \in T_{\Sigma_G} \mid S_G \Rightarrow_G^* t \,\}$.

4.2 Regular dependency grammars

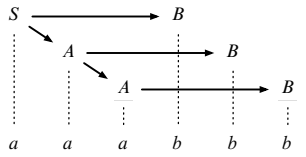
We call a dependency language *regular*, if its encoding as a set of trees over $\Sigma \times \Omega$ forms a regular tree language, and write REGD for the class of all regular dependency languages. For every regular dependency language L, there is a regular tree grammar with terminal alphabet $\Sigma \times \Omega$ that generates the encoding of L. Similar to the situation with individual structures, the converse of this statement does not hold: the consistency properties mentioned above impose corresponding syntactic restrictions on the rules of grammars G that generate the encoding of L.

PROPERTY C1′: The ω-component of every production $A \to \langle \sigma, \omega \rangle (A_1 \cdots A_n)$ in G contains all and only symbols in the set $\{0\} \cup \{\, i \mid i \in [n] \,\}$.

PROPERTY C2′: For every non-terminal $X \in N_G$, there is a uniquely determined integer d_X such that for every production $A \to \langle \sigma, \omega \rangle (A_1 \cdots A_n)$ in G, d_{A_i} gives the number of occurrences of i in ω, d_A gives the number of components in ω, and $d_{S_G} = 1$.

It turns out that these properties are in fact sufficient to characterize the class of regular tree grammars that generate encodings of dependency languages. In but slight abuse of terminology, we will refer to such grammars as *regular dependency grammars*.

EXAMPLE. Figure 3 shows a regular tree grammar that generates a set of non-projective dependency structures with string language $\{\, a^n b^n \mid n \geq 1 \,\}$. □



$S \to \langle a, \langle 01 \rangle \rangle (B) \mid \langle a, \langle 0121 \rangle \rangle (A, B)$
$A \to \langle a, \langle 0, 1 \rangle \rangle (B) \mid \langle a, \langle 01, 21 \rangle \rangle (A, B)$
$B \to \langle b, \langle 0 \rangle \rangle$

Figure 3: A grammar for a language in REGD(1)

5 Structural constraints and formal power

In this section, we present our results on the generative capacity of regular dependency languages, linking them to a large class of mildly context-sensitive grammar formalisms.

5.1 Gap-restricted dependency languages

A dependency language L is called *gap-restricted*, if there is a constant $c_L \geq 0$ such that no structure in L has a gap-degree higher than c_L. It is plain to see that every regular dependency language is gap-restricted: the gap-degree of a structure is directly reflected in the number of components of its order annotations, and every regular dependency grammar makes use of only a finite number of these annotations. We write REGD(k) to refer to the class of regular dependency languages with a gap-degree bounded by k.

Linear Context-Free Rewriting Systems Gap-restricted dependency languages are closely related to Linear Context-Free Rewriting Systems (LCFRS) (Vijay-Shanker et al., 1987), a class of formal systems that generalizes several mildly context-sensitive grammar formalisms. An LCFRS consists of a regular tree grammar G and an interpretation of the terminal symbols of this grammar as linear, non-erasing functions into tuples of strings. By these functions, each tree in $L(G)$ can be evaluated to a string.

EXAMPLE. Here is an example for a function:

$$f(\langle x_1^1, x_1^2 \rangle, \langle x_2^1 \rangle) \;=\; \langle a x_1^1, x_2^1 x_1^2 \rangle$$

This function states that in order to compute the pair of strings that corresponds to a tree whose root node is labelled with the symbol f, one first has to compute the pair of strings corresponding to the first child

of the root node ($\langle x_1^1, x_1^2 \rangle$) and the single string corresponding to the second child ($\langle x_2^1 \rangle$), and then concatenate the individual components in the specified order, preceded by the terminal symbol a. □

We call a function *lexicalized*, if it contributes exactly one terminal symbol. In an LCFRS in which all functions are lexicalized, there is a one-to-one correspondence between the nodes in an evaluated tree and the positions in the string that the tree evaluates to. Therefore, tree and string implicitly form a dependency structure, and we can speak of the *dependency language* generated by a lexicalized LCFRS.

Equivalence We can prove that every regular dependency grammar can be transformed into a lexicalized LCFRS that generates the same dependency language, and vice versa. The basic insight in this proof is that every order annotation in a regular dependency grammar can be interpreted as a compact description of a function in the corresponding LCFRS. The number of components in the order-annotation, and hence, the gap-degree of the resulting dependency language, corresponds to the *fan-out* of the function: the highest number of components among the arguments of the function (Satta, 1992).[1] A technical difficulty is caused by the fact that LCFRS can swap components: $f(\langle x_1^1, x_1^2 \rangle) = \langle a x_1^2, x_1^1 \rangle$. This commutativity needs to be compiled out during the translation into a regular dependency grammar.

We write LLCFRL(k) for the class of all dependency languages generated by lexicalized LCFRS with a fan-out of at most k.

PROPOSITION 4. $\text{REGD}(k) = \text{LLCFRL}(k+1)$ □

In particular, the class REGD(0) of regular dependency languages over projective structures is exactly the class of dependency languages generated by lexicalized context-free grammars.

EXAMPLE. The gap-degree of the language generated by the grammar in Figure 3 is bounded by 1. The rules for the non-terminal A can be translated into the following functions of an equivalent LCFRS:

$$f_{\langle a,\langle 0,1\rangle\rangle}(\langle x_1^1 \rangle) = \langle a, x_1^1 \rangle$$
$$f_{\langle a,\langle 01,21\rangle\rangle}(\langle x_1^1, x_1^2 \rangle, \langle x_2^1 \rangle) = \langle a x_1^1, x_2^1 x_1^2 \rangle$$

The fan-out of these functions is 2. □

[1] More precisely, gap-degree = fan-out − 1.

5.2 Well-nested dependency languages

The absence of the substring $i \cdots j \cdots i \cdots j$ in the order annotations of well-nested dependency structures corresponds to a restriction to 'well-bracketed' compositions of sub-structures. This restriction is central to the formalism of Coupled-Context-Free Grammar (CCFG) (Hotz and Pitsch, 1996).

It is straightforward to see that every CCFG can be translated into an equivalent LCFRS. We can also prove that every LCFRS obtained from a regular dependency grammar with well-nested order annotations can be translated back into an equivalent CCFG. We write $\text{REGD}_{wn}(k)$ for the well-nested subclass of REGD(k), and LCCFL(k) for the class of all dependency languages generated by lexicalized CCFGs with a fan-out of at most k.

PROPOSITION 5. $\text{REGD}_{wn}(k) = \text{LCCFL}(k+1)$ □

As a special case, Coupled-Context-Free Grammars with fan-out 2 are equivalent to Tree Adjoining Grammars (TAGS) (Hotz and Pitsch, 1996). This enables us to generalize a previous result on the class of dependency structures generated by lexicalized TAGS (Bodirsky et al., 2005) to the class of generated dependency languages, LTAL.

PROPOSITION 6. $\text{REGD}_{wn}(1) = \text{LTAL}$ □

6 Conclusion

In this paper, we have presented a lexicalized reformulation of two structural constraints on non-projective dependency representations, and shown that combining dependency lexicons that satisfy these constraints with a regular means of syntactic composition yields classes of mildly context-sensitive dependency languages. Our results make a significant contribution to a better understanding of the relation between the phenomenon of non-projectivity and notions of formal power.

The close link between restricted forms of non-projective dependency languages and mildly context-sensitive grammar formalisms provides a promising starting point for future work. On the practical side, it should allow us to benefit from the experience in building parsers for mildly context-sensitive formalisms when addressing the task of efficient non-projective dependency parsing, at least in the frame-

work of grammar-driven parsing. This may eventually lead to a better trade-off between structural flexibility and computational efficiency than that obtained with current systems. On a more theoretical level, our results provide a basis for comparing a variety of formally rather distinct grammar formalisms with respect to the sets of dependency structures that they can generate. Such a comparison may be empirically more adequate than one based on traditional notions of generative capacity (Kallmeyer, 2006).

Acknowledgements We thank Guido Tack, Stefan Thater, and the anonymous reviewers of this paper for their detailed comments. The work of the authors is funded by the German Research Foundation.

References

Manuel Bodirsky, Marco Kuhlmann, and Mathias Möhl. 2005. Well-nested drawings as models of syntactic structure. In *Tenth Conference on Formal Grammar and Ninth Meeting on Mathematics of Language*, Edinburgh, Scotland, UK.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 423–429, Barcelona, Spain.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 457–464, College Park, Maryland, USA.

Ferenc Gécseg and Magnus Steinby. 1997. Tree languages. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 1–68. Springer-Verlag, New York, USA.

Günter Hotz and Gisela Pitsch. 1996. On parsing coupled-context-free languages. *Theoretical Computer Science*, 161:205–233.

Aravind K. Joshi. 1985. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In David R. Dowty, Lauri Karttunen, and Arnold M. Zwicky, editors, *Natural Language Parsing*, pages 206–250. Cambridge University Press, Cambridge, UK.

Laura Kallmeyer. 2006. Comparing lexicalized grammar formalisms in an empirically adequate way: The notion of generative attachment capacity. In *International Conference on Linguistic Evidence*, pages 154–156, Tübingen, Germany.

Marco Kuhlmann and Joakim Nivre. 2006. Mildly non-projective dependency structures. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL) Main Conference Poster Sessions*, pages 507–514, Sydney, Australia.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Eleventh Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 81–88, Trento, Italy.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Human Language Technology Conference (HLT) and Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–530, Vancouver, British Columbia, Canada.

Peter Neuhaus and Norbert Bröker. 1997. The complexity of recognition of linguistically adequate dependency grammars. In *35th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 337–343, Madrid, Spain.

Joakim Nivre. 2006. Constraints on non-projective dependency parsing. In *Eleventh Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 73–80, Trento, Italy.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal smt. In *43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 271–279, Ann Arbor, USA.

Owen Rambow and Aravind K. Joshi. 1997. A formal look at dependency grammars and phrase-structure grammars. In Leo Wanner, editor, *Recent Trends in Meaning-Text Theory*, volume 39 of *Studies in Language, Companion Series*, pages 167–190. John Benjamins, Amsterdam, The Netherlands.

Giorgio Satta. 1992. Recognition of linear context-free rewriting systems. In *30th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 89–95, Newark, Delaware, USA.

Katerina Veselá, Jiři Havelka, and Eva Hajičova. 2004. Condition of projectivity in the underlying dependency structures. In *20th International Conference on Computational Linguistics (COLING)*, pages 289–295, Geneva, Switzerland.

K. Vijay-Shanker, David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *25th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 104–111, Stanford, California, USA.

Transforming Projective Bilexical Dependency Grammars into efficiently-parsable CFGs with Unfold-Fold

Mark Johnson

Microsoft Research
Redmond, WA
t-majoh@microsoft.com

Brown University
Providence, RI
Mark_Johnson@Brown.edu

Abstract

This paper shows how to use the Unfold-Fold transformation to transform Projective Bilexical Dependency Grammars (PBDGs) into ambiguity-preserving weakly equivalent Context-Free Grammars (CFGs). These CFGs can be parsed in $O(n^3)$ time using a CKY algorithm with appropriate indexing, rather than the $O(n^5)$ time required by a naive encoding. Informally, using the CKY algorithm with such a CFG mimics the steps of the Eisner-Satta $O(n^3)$ PBDG parsing algorithm. This transformation makes all of the techniques developed for CFGs available to PBDGs. We demonstrate this by describing a maximum posterior parse decoder for PBDGs.

1 Introduction

Projective Bilexical Dependency Grammars (PBDGs) have attracted attention recently for two reasons. First, because they capture bilexical head-to-head dependencies they are capable of producing extremely high-quality parses: state-of-the-art discriminatively trained PBDG parsers rival the accuracy of the very best statistical parsers available today (McDonald, 2006). Second, Eisner-Satta $O(n^3)$ PBDG parsing algorithms are extremely fast (Eisner, 1996; Eisner and Satta, 1999; Eisner, 2000).

This paper investigates the relationship between Context-Free Grammar (CFG) parsing and the Eisner/Satta PBDG parsing algorithms, including their extension to second-order PBDG parsing (McDonald, 2006; McDonald and Pereira, 2006). Specifically, we show how to use an off-line preprocessing step, the Unfold-Fold transformation, to transform a PBDG into an equivalent CFG that can be parsed in $O(n^3)$ time using a version of the CKY algorithm with suitable indexing (Younger, 1967), and extend this transformation so that it captures second-order PBDG dependencies as well. The transformations are ambiguity-preserving, i.e., there is a one-to-one mapping between dependency parses and CFG parses, so it is possible to map the CFG parses back to the PBDG parses they correspond to.

The PBDG to CFG reductions make techniques developed for CFGs available to PBDGs as well. For example, incremental CFG parsing algorithms can be used with the CFGs produced by this transform, as can the Inside-Outside estimation algorithm (Lari and Young, 1990) and more exotic methods such as estimating adjoined hidden states (Matsuzaki et al., 2005; Petrov et al., 2006). As an example application, we describe a maximum posterior parse decoder for PBDGs in Section 8.

The Unfold-Fold transformation is a calculus for transforming functional and logic programs into equivalent but (hopefully) faster programs (Burstall and Darlington, 1977). We use it here to transform CFGs encoding dependency grammars into other CFGs that are more efficiently parsable. Since CFGs can be expressed as Horn-clause logic programs (Pereira and Shieber, 1987) and the Unfold-Fold transformation is provably correct for such programs (Sato, 1992; Pettorossi and Proeitti, 1992), it follows that its application to CFGs is provably correct as well. The Unfold-Fold transformation is used here to derive the CFG schemata presented in sections 5–7. A system that uses these schemata (such as the one described in section 8) can implement

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 168–175,
Prague, Czech Republic, June 2007.

these schemata directly, so the Unfold-Fold transformation plays a theoretical role in this work, justifying the resulting CFG schemata.

The closest related work we are aware of is McAllester (1999), which also describes a reduction of PBDGs to efficiently-parsable CFGs and directly inspired this work. However, the CFGs produced by McAllester's transformation include epsilon-productions so they require a specialized CFG parsing algorithm, while the CFGs produced by the transformations described here have binary productions so they can be parsed with standard CFG parsing algorithms. Further, our approach extends to second-order PBDG parsing, while McAllester only discusses first-order PBDGs.

The rest of this paper is structured as follows. Section 2 defines projective dependency graphs and grammars and Section 3 reviews the "naive" encoding of PBDGs as CFGs with an $O(n^5)$ parse time, where n is the length of the string to be parsed. Section 4 introduces the "split-head" CFG encoding of PBDGs, which has an $O(n^4)$ parse time and serves as the input to the Unfold-Fold transform. Section 5 uses the Unfold-Fold transform to obtain a weakly-equivalent CFG encoding of PBDGs which can be parsed in $O(n^3)$ time, and presents timing results showing that the transformation does speed parsing. Sections 6 and 7 apply Unfold-Fold in slightly more complex ways to obtain CFG encodings of PBDGs that also make second-order dependencies available in $O(n^3)$ time parsable CFGs. Section 8 applies a PBDG to CFG transform to obtain a maximum posterior decoding parser for PBDGs.

2 Projective bilexical dependency parses and grammars

Let Σ be a finite set of $terminals$ (e.g., words), and let 0 be the *root terminal* not in Σ. If $w = (w_1, \ldots, w_n) \in \Sigma^\star$, let $w^\star = (0, w_1, \ldots, w_n)$, i.e., $w^\star$ is obtained by prefixing w with 0. A *dependency parse* G for w is a tree whose root is labeled 0 and whose other n vertices are labeled with each of the n terminals in w. If G contains an arc from u to v then we say that v is a *dependent* of u, and if G contains a path from u to v then we say that v is a *descendant* of u. If v is dependent of u that also precedes u in $w^\star$ then we say that v is a *left dependent* of u (right dependent and left and right descendants are defined similarly).

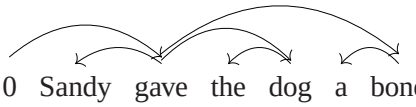


Figure 1: A projective dependency parse for the sentence "Sam gave the dog a bone".

A dependency parse G is *projective* iff whenever there is a path from u to v then there is also a path from u to every word between u and v in $w^\star$ as well. Figure 1 depicts a projective dependency parse for the sentence "Sam gave the dog a bone".

A projective dependency grammar defines a set of projective dependency parses. A *Projective Bilexical Dependency Grammar* (PBDG) consists of two relations $\overleftarrow{\ }$ and $\overrightarrow{\ }$, both defined over $(\Sigma \cup \{0\}) \times \Sigma$. A PBDG generates a projective dependency parse G iff $u \overrightarrow{\ } v$ for all right dependencies (u, v) in G and $v \overleftarrow{\ } u$ for all left dependencies (u, v) in G. The language generated by a PBDG is the set of strings that have projective dependency parses generated by the grammar. The following dependency grammar generates the dependency parse in Figure 1.

$$\begin{array}{cc} 0 \overrightarrow{\ } \text{gave} & \text{Sandy} \overleftarrow{\ } \text{gave} \\ \text{gave} \overrightarrow{\ } \text{dog} & \text{the} \overleftarrow{\ } \text{dog} \\ \text{gave} \overrightarrow{\ } \text{bone} & \text{a} \overleftarrow{\ } \text{bone} \end{array}$$

This paper does not consider stochastic dependency grammars directly, but see Section 8 for an application involving them. However, it is straightforward to associate weights with dependencies, and since the dependencies are preserved by the transformations, obtain a weighted CFG. Standard methods for converting weighted CFGs to equivalent PCFGs can be used if required (Chi, 1999). Alternatively, one can transform a corpus of dependency parses into a corpus of the corresponding CFG parses, and estimate CFG production probabilities directly from that corpus.

3 A naive encoding of PBDGs

There is a well-known method for encoding a PBDG as a CFG in which each terminal $u \in \Sigma$ is associated with a corresponding nonterminal X_u that expands to u and all of u's descendants. The nonterminals of the naive encoding CFG consist of the start symbol S and symbols X_u for each terminal $u \in \Sigma$, and

the productions of the CFG are the instances of the following schemata:

$$\begin{array}{rcll} \mathrm{S} & \rightarrow & \mathrm{X}_u & \text{where } 0 \curvearrowright u \\ \mathrm{X}_u & \rightarrow & u & \\ \mathrm{X}_u & \rightarrow & \mathrm{X}_v\ \mathrm{X}_u & \text{where } v \curvearrowleft u \\ \mathrm{X}_u & \rightarrow & \mathrm{X}_u\ \mathrm{X}_v & \text{where } u \curvearrowright v \end{array}$$

The dependency annotations associated with each production specify how to interpret a local tree generated by that production, and permit us to map a CFG parse to the corresponding dependency parse. For example, the top-most local tree in Figure 2 was generated by the production $\mathrm{S} \rightarrow \mathrm{X}_{\mathrm{gave}}$, and indicate that in this parse $0 \curvearrowright \mathrm{gave}$.

Given a terminal vocabulary of size m the CFG contains $O(m^2)$ productions, so it is impractical to enumerate all possible productions for even modest vocabularies. Instead productions relevant to a particular sentence are generated on the fly.

The naive encoding CFG in general requires $O(n^5)$ parsing time with a conventional CKY parsing algorithm, since tracking the head annotations u and v multiplies the standard $O(n^3)$ CFG parse time requirements by an additional factor proportional to the $O(n^2)$ productions expanding X_u.

An additional problem with the naive encoding is that the resulting CFG in general exhibits spurious ambiguities, i.e., a single dependency parse may correspond to more than one CFG parse, as shown in Figure 2. Informally, this is because the CFG permits left and the right dependencies to be arbitrarily intermingled.

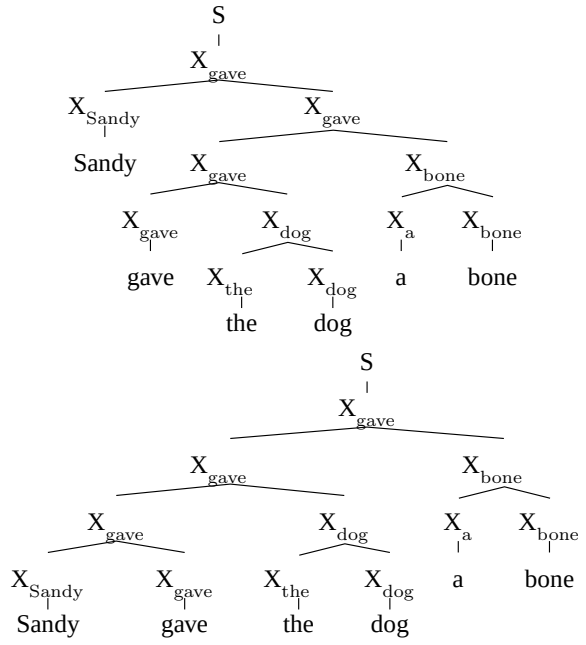


Figure 2: Two parses using the naive CFG encoding that both correspond to the dependency parse of Figure 1.

4 Split-head encoding of PBDGs

There are several ways of removing the spurious ambiguities in the naive CFG encoding just described. This section presents a method we call the "split-head encoding", which removes the ambiguities and serves as starting point for the grammar transforms described below.

The split-head encoding represents each word u in the input string w by *two* unique terminals u_l and u_r in the CFG parse. A split-head CFG's terminal vocabulary is $\Sigma' = \{u_l, u_r : u \in \Sigma\}$, where Σ is the set of terminals of the PBDG. A PBDG parse with yield $w = (u_1, \ldots, u_n)$ is transformed to a split-head CFG parse with yield $w' = (u_{1,l}, u_{1,r}, \ldots, u_{n,l}, u_{n,r})$, so $|w'| = 2|w|$.

The split-head CFG for a PBDG is given by the following schemata:

$$\begin{array}{rcll} \mathrm{S} & \rightarrow & \mathrm{X}_u & \text{where } 0 \curvearrowright u \\ \mathrm{X}_u & \rightarrow & \mathrm{L}_u\ {}_u\mathrm{R} & \text{where } u \in \Sigma \\ \mathrm{L}_u & \rightarrow & u_l & \\ \mathrm{L}_u & \rightarrow & \mathrm{X}_v\ \mathrm{L}_u & \text{where } v \curvearrowleft u \\ {}_u\mathrm{R} & \rightarrow & u_r & \\ {}_u\mathrm{R} & \rightarrow & {}_u\mathrm{R}\ \mathrm{X}_v & \text{where } u \curvearrowright v \end{array}$$

The dependency parse shown in Figure 1 corresponds to the split-head CFG parse shown in Figure 3. Each X_u expands to two new categories, L_u and ${}_u\mathrm{R}$. L_u consists of u_l and all of u's left descendants, while ${}_u\mathrm{R}$ consists of u_r and all of u's right descendants. The spurious ambiguity present in the naive encoding does not arise in the split-head encoding because the left and right dependents of a head are assembled independently and cannot intermingle.

As can be seen by examining the split-head schemata, the *rightmost* descendant of L_u is either L_u or u_l, which guarantees that the rightmost terminal dominated by L_u is always u_l; similarly the *leftmost* terminal dominated by ${}_u\mathrm{R}$ is always u_r. Thus

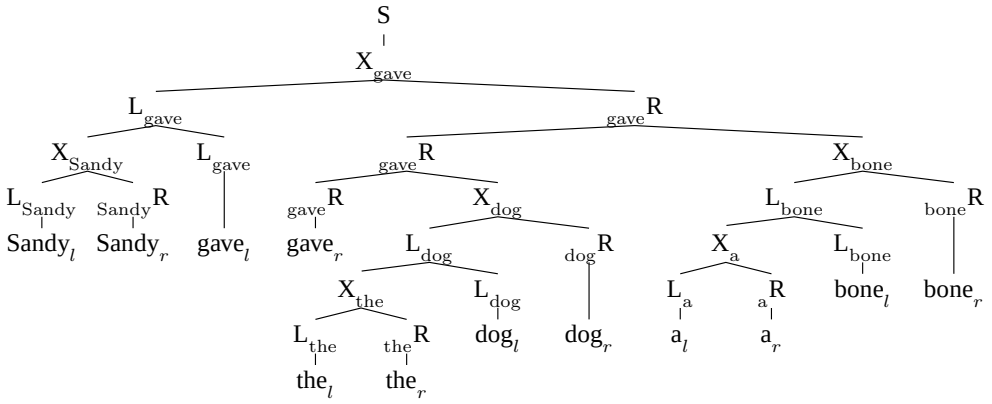


Figure 3: The split-head parse corresponding to the dependency graph depicted in Figure 1. Notice that u_l is always the rightmost descendant of L_u and u_r is always the leftmost descendant of ${}_u\mathrm{R}$, which means that these indices are redundant given the constituent spans.

these subscript indices are redundant given the string positions of the constituents, which means we do not need to track the index u in L_u and ${}_u\mathrm{R}$ but can parse with just the two categories L and R, and determine the index from the constituent's span when required.

It is straight-forward to extend the split-head CFG to encode the additional state information required by the head automata of Eisner and Satta (1999); this corresponds to splitting the non-terminals L_u and ${}_u\mathrm{R}$. For simplicity we work with PBDGs in this paper, but all of the Unfold-Fold transformations described below extend to split-head grammars with the additional state structure required by head automata.

Implementation note: it is possible to directly parse the "undoubled" input string w by modifying both the CKY algorithm and the CFGs described in this paper. Modify L_u and ${}_u\mathrm{R}$ so they both ultimately expand to the same terminal u, and special-case the implementation of production $\mathrm{X}_u \rightarrow \mathrm{L}_u\ {}_u\mathrm{R}$ and all productions derived from it to permit L_u and ${}_u\mathrm{R}$ to overlap by the terminal u.

The split-head formulation explains what initially seem unusual properties of existing PBDG algorithms. For example, one of the standard "sanity checks" for the Inside-Outside algorithm—that the outside probability of each terminal is equal to the sentence's inside probability—fails for these algorithms. In fact, the outside probability of each terminal is *double* the sentence's inside probability because these algorithms implicitly collapse the two terminals u_l and u_r into a single terminal u.

5 A $O(n^3)$ split-head grammar

The split-head encoding described in the previous section requires $O(n^4)$ parsing time because the index v on X_v is not redundant. We can obtain an equivalent grammar that only requires $O(n^3)$ parsing time by transforming the split-head grammar using Unfold-Fold. We describe the transformation on L_u; the transformation of ${}_u\mathrm{R}$ is symmetric.

We begin with the definition of L_u in the split-head grammar above ("|" separates the right-hand sides of productions).

$$\mathrm{L}_u \;\rightarrow\; u_l \;\mid\; \mathrm{X}_v\ \mathrm{L}_u \quad \text{where } v \curvearrowleft u$$

Our first transformation step is to unfold X_v in L_u, i.e., replace X_v by its expansion, producing the following definition for L_u (ignore the underlining for now).

$$\mathrm{L}_u \;\rightarrow\; u_l \;\mid\; \mathrm{L}_v\ \underline{{}_v\mathrm{R}\ \mathrm{L}_u} \quad \text{where } v \curvearrowleft u$$

This removes the offending X_v in L_u, but the resulting definition of L_u contains ternary productions and so still incurs $O(n^4)$ parse time. To address this we define new nonterminals ${}_x\mathrm{M}_y$ for each $x, y \in \Sigma$:

$${}_x\mathrm{M}_y \;\rightarrow\; {}_x\mathrm{R}\ \mathrm{L}_y$$

and fold the underlined children in L_u into ${}_v\mathrm{M}_u$:

$$\begin{aligned} {}_x\mathrm{M}_y &\rightarrow {}_x\mathrm{R}\ \mathrm{L}_y && \text{where } x, y \in \Sigma \\ \mathrm{L}_u &\rightarrow u_l \mid \mathrm{L}_v\ {}_v\mathrm{M}_u && \text{where } v \curvearrowleft u \end{aligned}$$

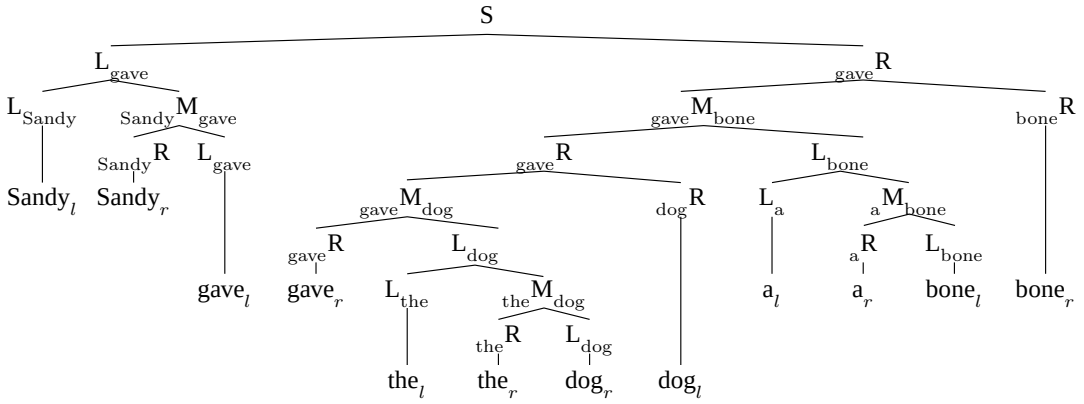


Figure 4: The $O(n^3)$ split-head parse corresponding to the dependency graph of Figure 1.

The $O(n^3)$ split-head grammar is obtained by unfolding the occurence of X_u in the S production and dropping the X_u schema as X_u no longer appears on the right-hand side of any production. The resulting $O(n^3)$ split-head grammar schemata are as follows:

$$
\begin{array}{rcll}
\mathrm{S} & \rightarrow & \mathrm{L}_u \ {}_u\mathrm{R} & \text{where } 0 \curvearrowright u \\
\mathrm{L}_u & \rightarrow & u_l & \\
\mathrm{L}_u & \rightarrow & \mathrm{L}_v \ {}_v\mathrm{M}_u & \text{where } v \curvearrowleft u \\
{}_u\mathrm{R} & \rightarrow & u_r & \\
{}_u\mathrm{R} & \rightarrow & {}_u\mathrm{M}_v \ {}_v\mathrm{R} & \text{where } u \curvearrowright v \\
{}_x\mathrm{M}_y & \rightarrow & {}_x\mathrm{R} \ \mathrm{L}_y & \text{where } x, y \in \Sigma
\end{array}
$$

As before, the dependency annotations on the production schemata permit us to map CFG parses to the corresponding dependency parse. This grammar requires $O(n^3)$ parsing time to parse because the indices are redundant given the constituent's string positions for the reasons described in section 4. Specifically, the rightmost terminal of L_u is always u_l, the leftmost terminal of ${}_u\mathrm{R}$ is always u_r and the leftmost and rightmost terminals of ${}_v\mathrm{M}_u$ are v_l and u_r respectively.

The $O(n^3)$ split-head grammar is closely related to the $O(n^3)$ PBDG parsing algorithm given by Eisner and Satta (1999). Specifically, the steps involved in parsing with this grammar using the CKY algorithm are essentially the same as those performed by the Eisner/Satta algorithm. The primary difference is that the Eisner/Satta algorithm involves two separate categories that are collapsed into the single category M here.

To confirm their relative performance we implemented stochastic CKY parsers for the three CFG schemata described so far. The production schemata were hard-coded for speed, and the implementation trick described in section 4 was used to avoid doubling the terminal string. We obtained dependency weights from our existing discriminatively-trained PBDG parser (not cited to preserve anonymity). We compared the parsers' running times on section 24 of the Penn Treebank. Because all three CFGs implement the same dependency grammar their Viterbi parses have the same dependency accuracy, namely 0.8918. We precompute the dependency weights, so the times include just the dynamic programming computation on a 3.6GHz Pentium 4.

CFG schemata	sentences parsed / second
Naive $O(n^5)$ CFG	45.4
$O(n^4)$ CFG	406.2
$O(n^3)$ CFG	3580.0

6 An $O(n^3)$ adjacent-head grammar

This section shows how to further transform the $O(n^3)$ grammar described above into a form that encodes second-order dependencies between adjacent dependent heads in much the way that a Markov PCFG does (McDonald, 2006; McDonald and Pereira, 2006). We provide a derivation for the L_u constituents; there is a parallel derivation for ${}_u\mathrm{R}$.

We begin by unfolding X_v in the definition of L_u in the split-head grammar, producing as before:

$$\mathrm{L}_u \ \rightarrow \ u_l \ | \ \mathrm{L}_v \ \underline{{}_v\mathrm{R} \ \mathrm{L}_u}$$

Now introduce a new nonterminal ${}_v\mathrm{M}_u^{\mathrm{L}}$, which is a specialized version of M requiring that v is a left-dependent of u, and fold the underlined constituents

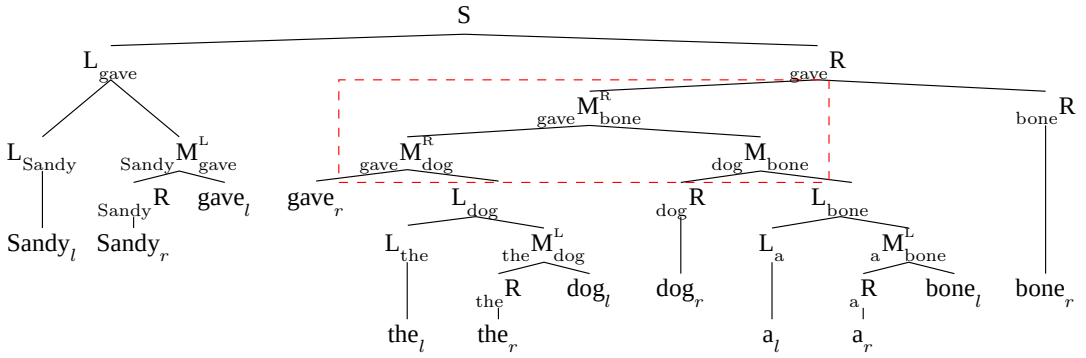


Figure 5: The $O(n^3)$ adjacent-head parse corresponding to the dependency graph of Figure 1. The boxed local tree indicates *bone* is the dependent of *give* following the dependent *dog*, i.e., $give \overset{\curvearrowright}{\curvearrowleft} dog \overset{\curvearrowright}{\curvearrowleft} bone$.

into ${}_v\mathrm{M}^{\mathrm{L}}_u$.

$$\begin{array}{rcll} {}_v\mathrm{M}^{\mathrm{L}}_u & \rightarrow & {}_v\mathrm{R}\ \mathrm{L}_u & \text{where } v \curvearrowleft u \\ \mathrm{L}_u & \rightarrow & u_l \mid \mathrm{L}_v\ {}_v\mathrm{M}^{\mathrm{L}}_u & \text{where } v \curvearrowleft u \end{array}$$

Now unfold L_u in the definition of ${}_v\mathrm{M}^{\mathrm{L}}_u$, producing:

$${}_v\mathrm{M}^{\mathrm{L}}_u \rightarrow {}_v\mathrm{R}\ u_l \mid \underline{{}_v\mathrm{R}\ \mathrm{L}_{v'}}\ {}_{v'}\mathrm{M}^{\mathrm{L}}_u; \quad v \overset{\curvearrowleft}{\curvearrowright} v' \overset{\curvearrowleft}{\curvearrowright} u$$

Note that in the first production expanding ${}_v\mathrm{M}^{\mathrm{L}}_u$, v is the *closest* left dependent of u, and in the second production v and v' are *adjacent* left-dependents of u. ${}_v\mathrm{M}^{\mathrm{L}}_u$ has a ternary production, so we introduce ${}_x\mathrm{M}_y$ as before to fold the underlined constituents into.

$$\begin{array}{rcll} {}_x\mathrm{M}_y & \rightarrow & {}_x\mathrm{R}\ \mathrm{L}_y & \text{where } x, y \in \Sigma \\ {}_v\mathrm{M}^{\mathrm{L}}_u & \rightarrow & {}_v\mathrm{R}\ u_l \mid {}_v\mathrm{M}_{v'}\ {}_{v'}\mathrm{M}^{\mathrm{L}}_u; & v \overset{\curvearrowleft}{\curvearrowright} v' \overset{\curvearrowleft}{\curvearrowright} u \end{array}$$

The resulting grammar schema is as below, and a sample parse is given in Figure 5.

$$\begin{array}{rcll} \mathrm{S} & \rightarrow & \mathrm{L}_u\ {}_u\mathrm{R} & \text{where } 0 \curvearrowright u \\ \mathrm{L}_u & \rightarrow & u_l & u \text{ has no left dependents} \\ \mathrm{L}_u & \rightarrow & \mathrm{L}_v\ {}_v\mathrm{M}^{\mathrm{L}}_u & v \text{ is } u\text{'s last left dep.} \\ {}_v\mathrm{M}^{\mathrm{L}}_u & \rightarrow & {}_v\mathrm{R}\ u_l & v \text{ is } u\text{'s closest left dep.} \\ {}_v\mathrm{M}^{\mathrm{L}}_u & \rightarrow & {}_v\mathrm{M}_{v'}\ {}_{v'}\mathrm{M}^{\mathrm{L}}_u & v \overset{\curvearrowleft}{\curvearrowright} v' \overset{\curvearrowleft}{\curvearrowright} u \\ {}_u\mathrm{R} & \rightarrow & u_r & u \text{ has no right dependents} \\ {}_u\mathrm{R} & \rightarrow & {}_u\mathrm{M}^{\mathrm{R}}_v\ {}_v\mathrm{R} & v \text{ is } u\text{'s last right dep.} \\ {}_u\mathrm{M}^{\mathrm{R}}_v & \rightarrow & u_r\ \mathrm{L}_v & v \text{ is } u\text{'s closest right dep.} \\ {}_u\mathrm{M}^{\mathrm{R}}_v & \rightarrow & {}_u\mathrm{M}^{\mathrm{R}}_{v'}\ {}_{v'}\mathrm{M}_v & u \overset{\curvearrowright}{\curvearrowleft} v' \overset{\curvearrowright}{\curvearrowleft} v \\ {}_x\mathrm{M}_y & \rightarrow & {}_x\mathrm{R}\ \mathrm{L}_y & \text{where } x, y \in \Sigma \end{array}$$

As before, the indices on the nonterminals are redundant, as the heads are always located at an edge of each constituent, so they need not be computed or stored and the CFG can be parsed in $O(n^3)$ time. The steps involved in CKY parsing with this grammar correspond closely to those of the McDonald (2006) second-order PBDG parsing algorithm.

7 An $O(n^3)$ dependent-head grammar

This section shows a different application of Unfold-Fold can capture head-to-head-to-head dependencies, i.e., "vertical" second-order dependencies, rather than the "horizontal" ones captured by the transformation described in the previous section. Because we expect these vertical dependencies to be less important linguistically than the horizontal ones, we only sketch the transformation here.

The derivation differs from the one in Section 6 in that the dependent ${}_v\mathrm{R}$, rather than the head L_u, is unfolded in the initial definition of ${}_v\mathrm{M}^{\mathrm{L}}_u$. This results in a grammar that tracks vertical, rather than horizontal, second-order dependencies. Since left-hand and right-hand derivations are assembled separately in a split-head grammar, the grammar in fact only tracks zig-zag type dependencies (e.g., where a grandparent has a right dependent, which in turn has a left dependent).

The resulting grammar is given below, and a sample parse using this grammar is shown in Figure 6. Because the subscripts are redundant they can be omitted and the resulting CFG can be parsed in

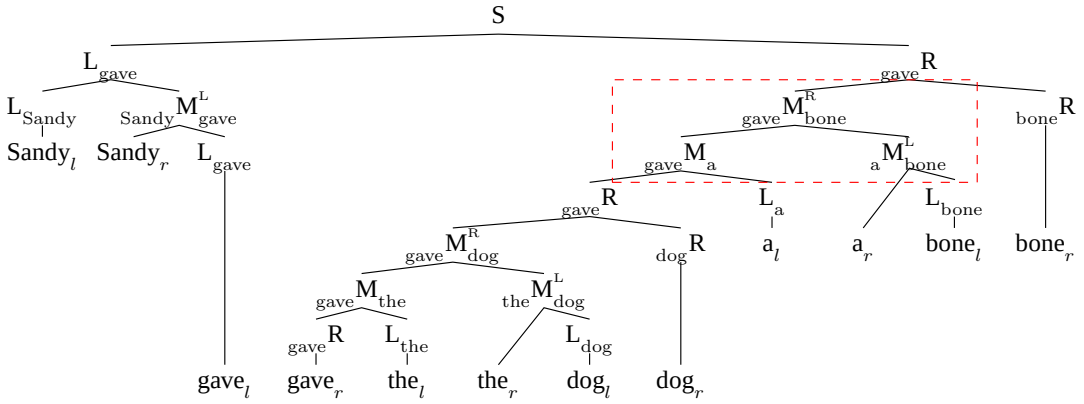


Figure 6: The n^3 dependent-head parse corresponding to the dependency graph of Figure 1. The boxed local tree indicates that *a* is a left-dependent of *bone*, which is in turn a right-dependent of *gave*, i.e., *gave* ← *a* → *bone* .

$O(n^3)$ time using the CKY algorithm.

$$
\begin{array}{rcll}
\mathrm{S} & \rightarrow & \mathrm{L}_{u\ u}\mathrm{R} & \text{where } 0 \curvearrowright u \\
\mathrm{L}_{u} & \rightarrow & u_l & \\
\mathrm{L}_{u} & \rightarrow & \mathrm{L}_{v\ v}\mathrm{M}^{\mathrm{L}}_{u} & \text{where } v \curvearrowleft u \\
{}_{v}\mathrm{M}^{\mathrm{L}}_{u} & \rightarrow & v_r\ \mathrm{L}_{u} & \text{where } v \curvearrowleft u \\
{}_{v}\mathrm{M}^{\mathrm{L}}_{u} & \rightarrow & {}_{v}\mathrm{M}^{\mathrm{R}}_{w\ w}\mathrm{M}_{u} & \text{where } v \leftrightarrows w \ u \\
{}_{u}\mathrm{R} & \rightarrow & u_r & \\
{}_{u}\mathrm{R} & \rightarrow & {}_{u}\mathrm{M}^{\mathrm{R}}_{v\ v}\mathrm{R} & \text{where } u \curvearrowright v \\
{}_{u}\mathrm{M}^{\mathrm{R}}_{v} & \rightarrow & {}_{u}\mathrm{R}\ v_l & \text{where } u \curvearrowright v \\
{}_{u}\mathrm{M}^{\mathrm{R}}_{v} & \rightarrow & {}_{u}\mathrm{M}_{w\ w}\mathrm{M}^{\mathrm{L}}_{v} & \text{where } u \ \overset{\curvearrowright}{\leftrightarrows}\ w \ u \\
{}_{x}\mathrm{M}_{y} & \rightarrow & {}_{x}\mathrm{R}\ \mathrm{L}_{y} & \text{where } x, y \in \Sigma
\end{array}
$$

8 Maximum posterior decoding

As noted in the introduction, one consequence of the PBDG to CFG reductions presented in this paper is that CFG parsing and estimation techniques are now available for PBDGs as well. As an example application, this section describes Maximum Posterior Decoding (MPD) for PBDGs.

Goodman (1996) observed that the Viterbi parse is in general not the optimal parse for evaluation metrics such as f-score that are based on the number of correct constituents in a parse. He showed that MPD improves f-score modestly relative to Viterbi decoding for PCFGs.

Since dependency parse accuracy is just the proportion of dependencies in the parse that are correct, Goodman's observation should hold for PBDG parsing as well. MPD for PBDGs selects the parse that maximizes the sum of the marginal probabilities of each of the dependencies in the parse. Such a decoder might plausibly produce parses that score better on the dependency accuracy metric than Viterbi parses.

MPD is straightforward given the PBDG to CFG reductions described in this paper. Specifically, we use the Inside-Outside algorithm to compute the posterior probability of the CFG constituents corresponding to each PBDG dependency, and then use the Viterbi algorithm to find the parse tree that maximizes the sum of these posterior probabilities.

We implemented MPD for first-order PBDGs using dependency weights from our existing discriminatively-trained PBDG parser (not cited to preserve anonymity). These weights are estimated by an online procedure as in McDonald (2006), and are not intended to define a probability distribution. In an attempt to heuristically correct for this, in this experiment we used $\exp(\alpha w_{u,v})$ as the weight of the dependency between head u and dependent v, where $w_{u,v}$ is the weight provided by the discriminatively-trained model and α is an adjustable scaling parameter tuned to optimize MPD accuracy on development data.

Unfortunately we found no significant difference between the accuracy of the MPD and Viterbi parses. Optimizing MPD on the development data (section 24 of the PTB) set the scale factor $\alpha = 0.21$ and produced MPD parses with an accuracy of 0.8921, which is approximately the same as the Viterbi accuracy of 0.8918. On the blind test data (section 23) the two accuracies are essentially iden-

tical (0.8997).

There are several possible explanations for the failure of MPD to produce more accurate parses than Viterbi decoding. Perhaps MPD requires weights that define a probability distribution (e.g., a MaxEnt model). It is also possible that discriminative training adjusts the weights in a way that ensures that the Viterbi parse is close to the maximum posterior parse. This was the case in our experiment, and if this is true with discriminative training in general, then maximum posterior decoding will not have much to offer to discriminative parsing.

9 Conclusion

This paper shows how to use the Unfold-Fold transform to translate PBDGs into CFGs that can be parsed in $O(n^3)$ time. A key component of this is the split-head construction, where each word u in the input is split into two terminals u_l and u_r of the CFG parse. We also showed how to systematically transform the split-head CFG into grammars which track second-order dependencies. We provided one grammar which captures horizontal second-order dependencies (McDonald, 2006), and another which captures vertical second-order head-to-head-to-head dependencies.

The grammars described here just scratch the surface of what is possible with Unfold-Fold. Notice that both of the second-order grammars have more nonterminals than the first-order grammar. If one is prepared to increase the number of nonterminals still further, it may be possible to track additional information about constituents (although if we insist on $O(n^3)$ parse time we will be unable to track the interaction of more than three heads at once).

References

R.M. Burstall and John Darlington. 1977. A transformation system for developing recursive programs. *Journal of the Association for Computing Machinery*, 24(1):44–67.

Zhiyi Chi. 1999. Statistical properties of probabilistic context-free grammars. *Computational Linguistics*, 25(1):131–160.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 457–480, University of Maryland.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING96: Proceedings of the 16th International Conference on Computational Linguistics*, pages 340–345, Copenhagen. Center for Sprogteknologi.

Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In Harry Bunt and Anton Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer Academic Publishers.

Joshua T. Goodman. 1996. Parsing algorithms and metrics. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 177–183, Santa Cruz, Ca.

K. Lari and S.J. Young. 1990. The estimation of Stochastic Context-Free Grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4(35-56).

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 75–82, Ann Arbor, Michigan, June. Association for Computational Linguistics.

David McAllester. 1999. A reformulation of Eisner and Sata's cubic time parser for split head automata grammars. Available from http://ttic.uchicago.edu/˜dmcallester/.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 81–88, Trento, Italy.

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

Fernando Pereira and Stuart M. Shieber. 1987. *Prolog and Natural Language Analysis*. Center for the Study of Language and Information, Stanford, CA.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July. Association for Computational Linguistics.

A. Pettorossi and M. Proeitti. 1992. Transformation of logic programs. In *Handbook of Logic in Artificial Intelligence*, volume 5, pages 697–787. Oxford University Press.

Taisuke Sato. 1992. Equivalence-preserving first-order unfold/fold transformation systems. *Theoretical Computer Science*, 105(1):57–84.

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

Parsing and Generation as Datalog Queries

Makoto Kanazawa
National Institute of Informatics
2–1–2 Hitotsubashi, Chiyoda-ku, Tokyo, 101–8430, Japan
kanazawa@nii.ac.jp

Abstract

We show that the problems of parsing and surface realization for grammar formalisms with "context-free" derivations, coupled with Montague semantics (under a certain restriction) can be reduced in a uniform way to Datalog query evaluation. As well as giving a polynomial-time algorithm for computing all derivation trees (in the form of a shared forest) from an input string or input logical form, this reduction has the following complexity-theoretic consequences for all such formalisms: (i) the decision problem of recognizing grammaticality (surface realizability) of an input string (logical form) is in LOGCFL; and (ii) the search problem of finding one logical form (surface string) from an input string (logical form) is in functional LOGCFL. Moreover, the generalized supplementary magic-sets rewriting of the Datalog program resulting from the reduction yields efficient Earley-style algorithms for both parsing and generation.

1 Introduction

The representation of context-free grammars (augmented with features) in terms of definite clause programs is well-known. In the case of a bare-bone CFG, the corresponding program is in the function-free subset of logic programming, known as *Datalog*. For example, determining whether a string John found a unicorn belongs to the language of the CFG in Figure 1 is equivalent to deciding whether the Datalog program in Figure 2 together with the *database* in (1) can derive the *query* "?– S(0, 4)."

(1) John(0, 1). found(1, 2). a(2, 3). unicorn(3, 4).

S → NP VP	V → found
VP → V NP	V → caught
V → V Conj V	Conj → and
NP → Det N	Det → a
NP → John	N → unicorn

Figure 1: A CFG.

S(i, j) :– NP(i, k), VP(k, j).	V(i, j) :– found(i, j).
VP(i, j) :– V(i, k), NP(k, j).	V(i, j) :– caught(i, j).
V(i, j) :– V(i, k), Conj(k, l), V(l, j).	Conj(i, j) :– and(i, j).
NP(i, j) :– Det(i, k), N(k, j).	Det(i, j) :– a(i, j).
NP(i, j) :– John(i, j).	N(i, j) :– unicorn(i, j).

Figure 2: The Datalog representation of a CFG.

By *naive* (or *seminaive*) bottom-up evaluation (see, e.g., Ullman, 1988), the answer to such a query can be computed in polynomial time in the size of the database for any Datalog program. By recording rule instances rather than derived facts, a packed representation of the complete set of Datalog derivation trees for a given query can also be obtained in polynomial time by the same technique. Since a Datalog derivation tree uniquely determines a grammar derivation tree, this gives a reduction of context-free recognition and parsing to query evaluation in Datalog.

In this paper, we show that a similar reduction to Datalog is possible for more powerful grammar formalisms with "context-free" derivations, such as *(multi-component) tree-adjoining grammars* (Joshi and Schabes, 1997; Weir, 1988), *IO macro grammars* (Fisher, 1968), and *(parallel) multiple context-free grammars* (Seki et al., 1991). For instance, the TAG in Figure 3 is represented by the Datalog program in Figure 4. Moreover, the method of reduc-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 176–183,
Prague, Czech Republic, June 2007.

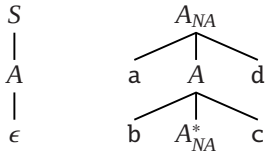


Figure 3: A TAG with one initial tree (left) and one auxiliary tree (right)

$$S(p_1, p_3) :- A(p_1, p_3, p_2, p_2).$$
$$A(p_1, p_8, p_4, p_5) :- A(p_2, p_7, p_3, p_6), \mathsf{a}(p_1, p_2), \mathsf{b}(p_3, p_4), \mathsf{c}(p_5, p_6), \mathsf{d}(p_7, p_8).$$
$$A(p_1, p_2, p_1, p_2).$$

Figure 4: The Datalog representation of a TAG.

tion extends to the problem of tactical generation (surface realization) for these grammar formalisms coupled with Montague semantics (under a certain restriction). Our method essentially relies on the encoding of different formalisms in terms of *abstract categorial grammars* (de Groote, 2001).

The reduction to Datalog makes it possible to apply to parsing and generation sophisticated evaluation techniques for Datalog queries; in particular, an application of *generalized supplementary magic-sets* rewriting (Beeri and Ramakrishnan, 1991) automatically yields Earley-style algorithms for both parsing and generation. The reduction can also be used to obtain a tight upper bound, namely *LOGCFL*, on the computational complexity of the problem of recognition, both for grammaticality of input strings and for surface realizability of input logical forms.

With regard to parsing and recognition of input strings, polynomial-time algorithms and the LOGCFL upper bound on the computational complexity are already known for the grammar formalisms covered by our results (Engelfriet, 1986); nevertheless, we believe that our reduction to Datalog offers valuable insights. Concerning generation, our results seem to be entirely new.[1]

2 Context-free grammars on λ-terms

Consider an augmentation of the grammar in Figure 1 with Montague semantics, where the left-hand side of each rule is annotated with a λ-term that tells how the meaning of the left-hand side is composed from the meanings of the right-hand side nonterminals, represented by upper-case variables $X_1, X_2, \ldots$ (Figure 5).[2]

$\mathsf{S}(X_1 X_2) \rightarrow \mathsf{NP}(X_1)\ \mathsf{VP}(X_2)$
$\mathsf{VP}(\lambda x.X_2(\lambda y.X_1 y x)) \rightarrow \mathsf{V}(X_1)\ \mathsf{NP}(X_2)$
$\mathsf{V}(\lambda y x.X_2(X_1 y x)(X_3 y x)) \rightarrow \mathsf{V}(X_1)\ \mathsf{Conj}(X_2)\ \mathsf{V}(X_3)$
$\mathsf{NP}(X_1 X_2) \rightarrow \mathsf{Det}(X_1)\ \mathsf{N}(X_2)$
$\mathsf{NP}(\lambda u.u\ \mathbf{John}^{e}) \rightarrow \mathsf{John}$
$\mathsf{V}(\mathbf{find}^{e\rightarrow e\rightarrow t}) \rightarrow \mathsf{found}$
$\mathsf{V}(\mathbf{catch}^{e\rightarrow e\rightarrow t}) \rightarrow \mathsf{caught}$
$\mathsf{Conj}(\wedge^{t\rightarrow t\rightarrow t}) \rightarrow \mathsf{and}$
$\mathsf{Det}(\lambda uv.\exists^{(e\rightarrow t)\rightarrow t}(\lambda y.\wedge^{t\rightarrow t\rightarrow t}(uy)(vy))) \rightarrow \mathsf{a}$
$\mathsf{N}(\mathbf{unicorn}^{e\rightarrow t}) \rightarrow \mathsf{unicorn}$

Figure 5: A context-free grammar with Montague semantics.

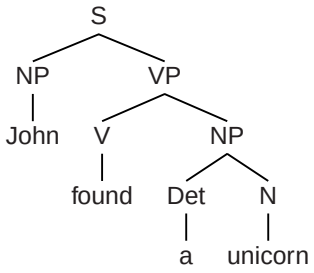


Figure 6: A derivation tree.

The meaning of a sentence is computed from its derivation tree. For example, John found a unicorn has the derivation tree in Figure 6, and the grammar rules assign its root node the λ-term

$$(\lambda u.u\ \mathbf{John})(\lambda x.(\lambda uv.\exists(\lambda y.\wedge(uy)(vy)))\ \mathbf{unicorn}\ (\lambda y.\mathbf{find}\ y\ x)),$$

which β-reduces to the λ-term

(2) $\exists(\lambda y.\wedge(\mathbf{unicorn}\ y)(\mathbf{find}\ y\ \mathbf{John}))$

encoding the first-order logic formula representing the meaning of the sentence (i.e., its logical form).

Thus, computing the logical form(s) of a sentence involves parsing and λ-term normalization. To find a sentence expressing a given logical form, it suffices

[1] We only consider *exact generation*, not taking into account the problem of logical form equivalence, which will most likely render the problem of generation computationally intractable (Moore, 2002).

[2] We follow standard notational conventions in typed λ-calculus. Thus, an application $M_1 M_2 M_3$ (written without parentheses) associates to the left, $\lambda x.\lambda y.M$ is abbreviated to $\lambda xy.M$, and $\alpha \rightarrow \beta \rightarrow \gamma$ stands for $\alpha \rightarrow (\beta \rightarrow \gamma)$. We refer the reader to Hindley, 1997 or Sørensen and Urzyczyn, 2006 for standard notions used in simply typed λ-calculus.

$\mathsf{S}(X_1X_2)$:– $\mathsf{NP}(X_1), \mathsf{VP}(X_2)$.
$\mathsf{VP}(\lambda x.X_2(\lambda y.X_1yx))$:– $\mathsf{V}(X_1), \mathsf{NP}(X_2)$.
$\mathsf{V}(\lambda yx.X_2(X_1yx)(X_3yx))$:– $\mathsf{V}(X_1), \mathsf{Conj}(X_2), \mathsf{V}(X_3)$.
$\mathsf{NP}(X_1X_2)$:– $\mathsf{Det}(X_1), \mathsf{N}(X_2)$.
$\mathsf{NP}(\lambda u.u\ \mathbf{John}^{e})$.
$\mathsf{V}(\mathbf{find}^{e\to e\to t})$.
$\mathsf{V}(\mathbf{catch}^{e\to e\to t})$.
$\mathsf{Conj}(\wedge^{t\to t\to t})$.
$\mathsf{Det}(\lambda uv.\exists^{(e\to t)\to t}(\lambda y.\wedge^{t\to t\to t}(uy)(vy)))$.
$\mathsf{N}(\mathbf{unicorn}^{e\to t})$.

Figure 7: A CFLG.

to find a derivation tree whose root node is associated with a λ-term that β-reduces to the given logical form; the desired sentence can simply be read off from the derivation tree. At the heart of both tasks is the computation of the derivation tree(s) that yield the input. In the case of generation, this may be viewed as parsing the input λ-term with a "context-free" grammar that generates a set of λ-terms (in normal form) (Figure 7), which is obtained from the original CFG with Montague semantics by stripping off terminal symbols. Determining whether a given logical form is surface realizable with the original grammar is equivalent to recognition with the resulting *context-free λ-term grammar* (CFLG).

In a CFLG such as in Figure 7, constants appearing in the λ-terms have preassigned types indicated by superscripts. There is a mapping σ from nonterminals to their types ($\sigma = \{\mathsf{S} \mapsto t, \mathsf{NP} \mapsto (e \to t) \to t, \mathsf{VP} \mapsto e{\to}t, \mathsf{V} \mapsto e{\to}e{\to}t, \mathsf{Conj} \mapsto t{\to}t{\to}t, \mathsf{Det} \mapsto (e{\to}t){\to}(e{\to}t){\to}t, \mathsf{N} \mapsto e{\to}t\}$). A rule that has A on the left-hand side and $B_1, \ldots, B_n$ as right-hand side nonterminals has its left-hand side annotated with a well-formed λ-term M that has type $\sigma(A)$ under the type environment $X_1 : \sigma(B_1), \ldots, X_n : \sigma(B_n)$ (in symbols, $X_1 : \sigma(B_1), \ldots, X_n : \sigma(B_n) \vdash M : \sigma(A)$).

What we have called a context-free λ-term grammar is nothing but an alternative notation for an *abstract categorial grammar* (de Groote, 2001) whose abstract vocabulary is second-order, with the restriction to *linear* λ-terms removed.[3] In the linear case, Salvati (2005) has shown the recognition/parsing complexity to be PTIME, and exhibited an algorithm similar to Earley parsing for TAGs. Second-order

[3] A λ-term is a *λI-term* if each occurrence of λ binds at least one occurrence of a variable. A λI-term is *linear* if no subterm contains more than one free occurrence of the same variable.

$S(\lambda y.X_1(\lambda z.z)y)$:– $A(X_1)$.
$A(\lambda xy.\mathsf{a}^{o\to o}(X_1(\lambda z.\mathsf{b}^{o\to o}(x(\mathsf{c}^{o\to o}z)))(\mathsf{d}^{o\to o}y)))$:– $A(X_1)$.
$A(\lambda xy.xy)$.

Figure 8: The CFLG encoding a TAG.

linear ACGs are known to be expressive enough to encode well-known mildly context-sensitive grammar formalisms in a straightforward way, including TAGs and multiple context-free grammars (de Groote, 2002; de Groote and Pogodalla, 2004).

For example, the linear CFLG in Figure 8 is an encoding of the TAG in Figure 3, where $\sigma(S) = o{\to}o$ and $\sigma(A) = (o \to o) \to o \to o$ (see de Groote, 2002 for details of this encoding). In encoding a string-generating grammar, a CFLG uses o as the type of string position and $o \to o$ as the type of string. Each terminal symbol is represented by a constant of type $o \to o$, and a string $a_1 \ldots a_n$ is encoded by the λ-term $\lambda z.a_1^{o\to o}(\ldots(a_n^{o\to o}z)\ldots)$, which has type $o \to o$.

A string-generating grammar coupled with Montague semantics may be represented by a *synchronous CFLG*, a pair of CFLGs with matching rule sets (de Groote 2001). The transduction between strings and logical forms in either direction consists of parsing the input λ-term with the source-side grammar and normalizing the λ-term(s) constructed in accordance with the target-side grammar from the derivation tree(s) output by parsing.

3 Reduction to Datalog

We show that under a weaker condition than linearity, a CFLG can be represented by a Datalog program, obtaining a tight upper bound (LOGCFL) on the recognition complexity. Due to space limitation, our presentation here is kept at an informal level; formal definitions and rigorous proof of correctness will appear elsewhere.

We use the grammar in Figure 7 as an example, which is represented by the Datalog program in Figure 9. Note that all λ-terms in this grammar are *almost linear* in the sense that they are λI-terms where any variable occurring free more than once in any subterm must have an atomic type. Our construction is guaranteed to be correct only when this condition is met.

Each Datalog rule is obtained from the corresponding grammar rule in the following way. Let

$\mathsf{S}(p_1)$:– $\mathsf{NP}(p_1, p_2, p_3), \mathsf{VP}(p_2, p_3)$.
$\mathsf{VP}(p_1, p_4)$:– $\mathsf{V}(p_2, p_4, p_3), \mathsf{NP}(p_1, p_2, p_3)$.
$\mathsf{V}(p_1, p_4, p_3)$:–
    $\mathsf{V}(p_2, p_4, p_3), \mathsf{Conj}(p_1, p_5, p_2), \mathsf{V}(p_5, p_4, p_3)$.
$\mathsf{NP}(p_1, p_4, p_5)$:– $\mathsf{Det}(p_1, p_4, p_5, p_2, p_3), \mathsf{N}(p_2, p_3)$.
$\mathsf{NP}(p_1, p_1, p_2)$:– $\mathbf{John}(p_2)$.
$\mathsf{V}(p_1, p_3, p_2)$:– $\mathbf{find}(p_1, p_3, p_2)$.
$\mathsf{V}(p_1, p_3, p_2)$:– $\mathbf{catch}(p_1, p_3, p_2)$.
$\mathsf{Conj}(p_1, p_3, p_2)$:– $\wedge(p_1, p_3, p_2)$.
$\mathsf{Det}(p_1, p_5, p_4, p_3, p_4)$:– $\exists(p_1, p_2, p_4), \wedge(p_2, p_5, p_3)$.
$\mathsf{N}(p_1, p_2)$:– $\mathbf{unicorn}(p_1, p_2)$.

Figure 9: The Datalog representation of a CFLG.

M be the λ-term annotating the left-hand side of the grammar rule. We first obtain a *principal* (i.e., most general) *typing* of M.[4] In the case of the second rule, this is

$$X_1 : p_3 \to p_4 \to p_2,\ X_2 : (p_3 \to p_2) \to p_1 \vdash \lambda x.X_2(\lambda y.X_1 y x) : p_4 \to p_1.$$

We then remove $\to$ and parentheses from the types in the principal typing and write the resulting sequences of atomic types in reverse.[5] We obtain the Datalog rule by replacing X_i and M in the grammar rule with the sequence coming from the type paired with X_i and M, respectively. Note that atomic types in the principal typing become variables in the Datalog rule. When there are constants in the λ-term M, they are treated like free variables. In the case of the second-to-last rule, the principal typing is

$$\exists : (p_4 \to p_2) \to p_1,\ \wedge : p_3 \to p_5 \to p_2 \vdash \lambda uv.\exists(\lambda y.\wedge(uy)(vy)) : (p_4 \to p_3) \to (p_4 \to p_5) \to p_1.$$

If the same constant occurs more than once, distinct occurrences are treated as distinct free variables.

The construction of the database representing the input λ-term is similar, but slightly more complex. A simple case is the λ-term (2), where each constant occurs just once. We compute its principal typing, treating constants as free variables.[6]

$$\exists : (4 \to 2) \to 1,\ \wedge : 3 \to 5 \to 2,\ \mathbf{unicorn} : 4 \to 3,\ \mathbf{find} : 4 \to 6 \to 5,\ \mathbf{John} : 6 \vdash \exists(\lambda y.\wedge(\mathbf{unicorn}\, y)(\mathbf{find}\, y\ \mathbf{John})) : 1.$$

[4]To be precise, we must first convert M to its *η-long form* relative to the type assigned to it by the grammar. For example, X_1X_2 in the first rule is converted to $X_1(\lambda x.X_2 x)$.

[5]The reason for reversing the sequences of atomic types is to reconcile the λ-term encoding of strings with the convention of listing string positions from left to right in databases like (1).

[6]We assume that the input λ-term is in η-long normal form.

We then obtain the corresponding database (3) and query (4) from the antecedent and succedent of this judgment, respectively. Note that here we are using $1, 2, 3, \ldots$ as atomic types, which become database constants.

(3) $\exists(1,2,4).\ \wedge(2,5,3).\ \mathbf{unicorn}(3,4).\ \mathbf{find}(5,6,4).\ \mathbf{John}(6).$

(4) ?– $\mathsf{S}(1)$.

When the input λ-term contains more than one occurrence of the same constant, it is not always correct to simply treat them as distinct free variables, unlike in the case of λ-terms annotating grammar rules. Consider the λ-term (5) (John found and caught a unicorn):

(5) $\exists(\lambda y.\wedge(\mathbf{unicorn}\, y)(\wedge(\mathbf{find}\, y\ \mathbf{John})(\mathbf{catch}\, y\ \mathbf{John})))$.

Here, the two occurrences of **John** must be treated as the same variable. The principal typing is (6) and the resulting database is (7).

(6) $\exists : (4 \to 2) \to 1,\ \wedge_1 : 3 \to 5 \to 2,\ \mathbf{unicorn} : 4 \to 3,\ \wedge_2 : 6 \to 8 \to 5,\ \mathbf{find} : 4 \to 7 \to 6,\ \mathbf{John} : 7, \mathbf{catch} : 4 \to 7 \to 8 \vdash \exists(\lambda y.\wedge_1(\mathbf{unicorn}\, y)(\wedge_2(\mathbf{find}\, y\ \mathbf{John})(\mathbf{catch}\, y\ \mathbf{John}))) : 1.$

(7) $\exists(1,2,4).\ \wedge(2,5,3).\ \wedge(5,8,6).\ \mathbf{unicron}(3,4).\ \mathbf{find}(6,7,4).\ \mathbf{John}(7).\ \mathbf{catch}(8,7,4).$

It is not correct to identify the two occurrences of $\wedge$ in this example. The rule is to identify distinct occurrences of the same constant just in case they occur in the same position within α-equivalent subterms of an atomic type. This is a necessary condition for those occurrences to originate as one and the same occurrence in the non-normal λ-term at the root of the derivation tree. (As a preprocessing step, it is also necessary to check that distinct occurrences of a bound variable satisfy the same condition, so that the given λ-term is β-equal to some almost linear λ-term.[7])

[7]Note that the way we obtain a database from an input λ-term generalizes the standard database representation of a string: from the λ-term encoding $\lambda z.a_1^{o\to o}(\ldots(a_n^{o\to o} z)\ldots)$ of a string $a_1 \ldots a_n$, we obtain the database $\{a_1(0,1), \ldots, a_n(n-1,n)\}$.

4 Correctness of the reduction

We sketch some key points in the proof of correctness of our reduction. The λ-term N obtained from the input λ-term by replacing occurrences of constants by free variables in the manner described above is the normal form of some almost linear λ-term N'. The *leftmost reduction* from an almost linear λ-term to its normal form must be *non-deleting* and *almost non-duplicating* in the sense that when a β-redex $(\lambda x.P)Q$ is contracted, Q is not deleted, and moreover it is not duplicated unless the type of x is atomic. We can show that the *Subject Expansion Theorem* holds for such β-reduction, so the principal typing of N is also the principal typing of N'. By a slight generalization of a result by Aoto (1999), this typing $\Gamma \vdash N' : \alpha$ must be *negatively non-duplicated* in the sense that each atomic type has at most one negative occurrence in it. By Aoto and Ono's (1994) generalization of the *Coherence Theorem* (see Mints, 2000), it follows that every λ-term P such that $\Gamma' \vdash P : \alpha$ for some $\Gamma' \subseteq \Gamma$ must be $\beta\eta$-equal to N' (and consequently to N).

Given the one-one correspondence between the grammar rules and the Datalog rules, a Datalog derivation tree uniquely determines a grammar derivation tree (see Figure 10 as an example). This relation is not one-one, because a Datalog derivation tree contains database constants from the input database. This extra information determines a typing of the λ-term P at the root of the grammar derivation tree (with occurrences of constants in the λ-term corresponding to distinct facts in the database regarded as distinct free variables):

$\mathbf{John} : 6$, $\mathbf{find} : 4 \to 6 \to 5$, $\exists : (4 \to 2) \to 1$,
$\wedge : 3 \to 5 \to 2$, $\mathbf{unicorn} : 4 \to 3 \vdash$
$(\lambda u.u\ \mathbf{John})$
$(\lambda x.(\lambda uv.\exists(\lambda y.\wedge(uy)(vy)))\ \mathbf{unicorn}\ (\lambda y.\mathbf{find}\ y\ x)) : 1.$

The antecedent of this typing must be a subset of the antecedent of the principal typing of the λ-term N from which the input database was obtained. By the property mentioned at the end of the preceding paragraph, it follows that the grammar derivation tree is a derivation tree for the input λ-term.

Conversely, consider the λ-term P (with distinct occurrences of constants regarded as distinct free variables) at the root of a grammar derivation tree for the input λ-term. We can show that there is a substitution θ which maps the free variables of P to the free variables of the λ-term N used to build the input database such that θ sends the normal form of P to N. Since P is an almost linear λ-term, the leftmost reduction from $P\theta$ to N is non-deleting and almost non-duplicating. By the Subject Expansion Theorem, the principal typing of N is also the principal typing of $P\theta$, and this together with the grammar derivation tree determines a Datalog derivation tree.

5 Complexity-theoretic consequences

Let us call a rule $A(M) :- B_1(X_1), \ldots, B_n(X_n)$ in a CFLG an *ϵ-rule* if $n = 0$ and M does not contain any constants. We can eliminate ϵ-rules from an almost linear CFLG by the same method that Kanazawa and Yoshinaka (2005) used for linear grammars, noting that for any Γ and α, there are only finitely many almost linear λ-terms M such that $\Gamma \vdash M : \alpha$. If a grammar has no ϵ-rule, any derivation tree for the input λ-term N that has a λ-term P at its root node corresponds to a Datalog derivation tree whose number of leaves is equal to the number of occurrences of constants in P, which cannot exceed the number of occurrences of constants in N.

A Datalog program $\mathbf{P}$ is said to have the *polynomial fringe property* relative to a class $\mathcal{D}$ of databases if there is a polynomial $p(n)$ such that for every database D in $\mathcal{D}$ of n facts and every query q such that $\mathbf{P} \cup D$ derives q, there is a derivation tree for q whose fringe (i.e., sequence of leaves) is of length at most $p(n)$. For such $\mathbf{P}$ and $\mathcal{D}$, it is known that $\{ (D, q) \mid D \in \mathcal{D}, \mathbf{P} \cup D \text{ derives } q \}$ is in the complexity class *LOGCFL* (Ullman and Van Gelder, 1988; Kanellakis, 1988).

We state without proof that the database-query pair (D, q) representing an input λ-term N can be computed in logspace. By padding D with extra useless facts so that the size of D becomes equal to the number of occurrences of constants in N, we obtain a logspace reduction from the set of λ-terms generated by an almost linear CFLG to a set of the form $\{ (D, q) \mid D \in \mathcal{D}, \mathbf{P} \cup D \vdash q \}$, where $\mathbf{P}$ has the polynomial fringe property relative to $\mathcal{D}$. This shows that the problem of recognition for an almost linear CFLG is in LOGCFL.

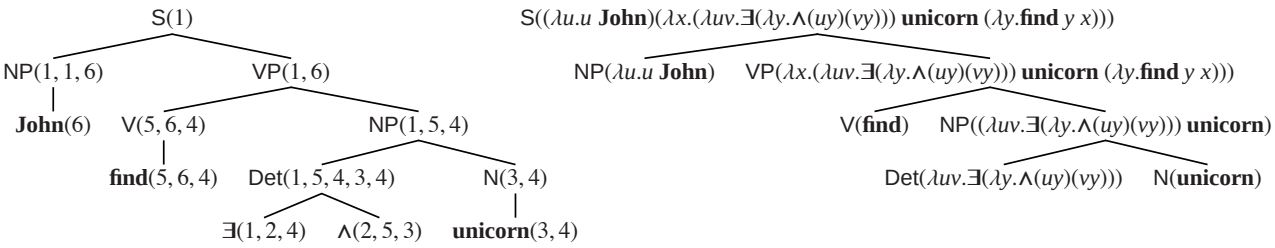


Figure 10: A Datalog derivation tree (left) and the corresponding grammar derivation tree (right)

By the main result of Gottlob et al. (2002), the related search problem of finding one derivation tree for the input λ-term is in *functional LOGCFL*, i.e., the class of functions that can be computed by a logspace-bounded Turing machine with a LOGCFL oracle. In the case of a synchronous almost linear CFLG, the derivation tree found from the source λ-term can be used to compute a target λ-term. Thus, to the extent that transduction back and forth between strings and logical forms can be expressed by a synchronous almost linear CFLG, the search problem of finding one logical form of an input sentence and that of finding one surface realization of an input logical form are both in functional LOGCFL.[8] As a consequence, there are efficient parallel algorithms for these problems.

6 Regular sets of trees as input

Almost linear CFLGs can represent a substantial fragment of a Montague semantics for English and such "linear" grammar formalisms as (multi-component) tree-adjoining grammars (both as string grammars and as tree grammars) and multiple context-free grammars. However, IO macro grammars and parallel multiple context-free grammars cannot be directly represented because representing string copying requires multiple occurrences of a variable of type $o \to o$. This problem can be solved by switching from strings to trees. We convert the input string into the regular set of binary trees whose yield equals the input string (using c as the sole symbol of rank 2), and turn the grammar into a tree grammar, replacing all instances of string concatenation in the grammar with the tree operation $t_1, t_2 \mapsto c(t_1, t_2)$. This way, a string grammar is turned into a tree grammar that generates a set of trees whose image under the yield function is the language of the string grammar. (In the case of an IO macro grammar, the result is an *IO context-free tree grammar* (Engelfriet, 1977).) String copying becomes tree copying, and the resulting grammar can be represented by an almost linear CFLG and hence by a Datalog program. The regular set of all binary trees that yield the input string is represented by a database that is constructed from a deterministic bottom-up finite tree automaton recognizing it. Determinism is important for ensuring correctness of this reduction. Since the database can be computed from the input string in logspace, the complexity-theoretic consequences of the last section carry over here.

7 Magic sets and Earley-style algorithms

Magic-sets rewriting of a Datalog program allows bottom-up evaluation to avoid deriving useless facts by mimicking top-down evaluation of the original program. The result of the *generalized supplementary magic-sets* rewriting of Beeri and Ramakrishnan (1991) applied to the Datalog program representing a CFG essentially coincides with the *deduction system* (Shieber et al., 1995) or *uninstantiated parsing system* (Sikkel, 1997) for Earley parsing. By applying the same rewriting method to Datalog programs representing almost linear CFLGs, we can obtain efficient parsing and generation algorithms for various grammar formalisms with context-free derivations.

We illustrate this approach with the program in Figure 4, following the presentation of Ullman

[8] If the target-side grammar is not linear, the normal form of the target λ-term cannot be explicitly computed because its size may be exponential in the size of the source λ-term. Nevertheless, a typing that serves to uniquely identify the target λ-term can be computed from the derivation tree in logspace. Also, if the target-side grammar is linear and string-generating, the target string can be explicitly computed from the derivation tree in logspace (Salvati, 2007).

(1989a; 1989b). We assume the query to take the form "?– $S(0, x)$.", so that the input database can be processed incrementally. The program is first made *safe* by eliminating the possibility of deriving non-ground atoms:

$S(p_1, p_3)$:– $A(p_1, p_3, p_2, p_2)$.
$A(p_1, p_8, p_4, p_5)$:– $A(p_2, p_7, p_3, p_6)$, $\mathsf{a}(p_1, p_2)$, $\mathsf{b}(p_3, p_4)$, $\mathsf{c}(p_5, p_6)$, $\mathsf{d}(p_7, p_8)$.
$A(p_1, p_8, p_4, p_5)$:– $\mathsf{a}(p_1, p_2)$, $\mathsf{b}(p_2, p_4)$, $\mathsf{c}(p_5, p_6)$, $\mathsf{d}(p_6, p_8)$.

The *subgoal rectification* removes duplicate arguments from subgoals, creating new predicates as needed:

$S(p_1, p_3)$:– $B(p_1, p_3, p_2)$.
$A(p_1, p_8, p_4, p_5)$:– $A(p_2, p_7, p_3, p_6)$, $\mathsf{a}(p_1, p_2)$, $\mathsf{b}(p_3, p_4)$, $\mathsf{c}(p_5, p_6)$, $\mathsf{d}(p_7, p_8)$.
$A(p_1, p_8, p_4, p_5)$:– $\mathsf{a}(p_1, p_2)$, $\mathsf{b}(p_2, p_4)$, $\mathsf{c}(p_5, p_6)$, $\mathsf{d}(p_6, p_8)$.
$B(p_1, p_8, p_4)$:– $A(p_2, p_7, p_3, p_6)$, $\mathsf{a}(p_1, p_2)$, $\mathsf{b}(p_3, p_4)$, $\mathsf{c}(p_4, p_6)$, $\mathsf{d}(p_7, p_8)$.
$B(p_1, p_8, p_4)$:– $\mathsf{a}(p_1, p_2)$, $\mathsf{b}(p_2, p_4)$, $\mathsf{c}(p_4, p_6)$, $\mathsf{d}(p_6, p_8)$.

We then attach to predicates *adornments* indicating the free/bound status of arguments in top-down evaluation, reordering subgoals so that as many arguments as possible are marked as bound:

$S^{bf}(p_1, p_3)$:– $B^{bff}(p_1, p_3, p_2)$.
$B^{bff}(p_1, p_8, p_4)$:– $\mathsf{a}^{bf}(p_1, p_2)$, $A^{bfff}(p_2, p_7, p_3, p_6)$, $\mathsf{b}^{bf}(p_3, p_4)$, $\mathsf{c}^{bb}(p_4, p_6)$, $\mathsf{d}^{bf}(p_7, p_8)$.
$B^{bff}(p_1, p_8, p_4)$:– $\mathsf{a}^{bf}(p_1, p_2)$, $\mathsf{b}^{bf}(p_2, p_4)$, $\mathsf{c}^{bf}(p_4, p_6)$, $\mathsf{d}^{bf}(p_6, p_8)$.
$A^{bfff}(p_1, p_8, p_4, p_5)$:– $\mathsf{a}^{bf}(p_1, p_2)$, $A^{bfff}(p_2, p_7, p_3, p_6)$, $\mathsf{b}^{bf}(p_3, p_4)$, $\mathsf{c}^{bb}(p_5, p_6)$, $\mathsf{d}^{bf}(p_7, p_8)$.
$A^{bfff}(p_1, p_8, p_4, p_5)$:– $\mathsf{a}^{bf}(p_1, p_2)$, $\mathsf{b}^{bf}(p_2, p_4)$, $\mathsf{c}^{ff}(p_5, p_6)$, $\mathsf{d}^{bf}(p_6, p_8)$.

The generalized supplementary magic-sets rewriting finally gives the following rule set:

r_1 : $m\_B(p_1)$:– $m\_S(p_1)$.
r_2 : $S(p_1, p_3)$:– $m\_B(p_1)$, $B(p_1, p_3, p_2)$.
r_3 : $sup_{2.1}(p_1, p_2)$:– $m\_B(p_1)$, $\mathsf{a}(p_1, p_2)$.
r_4 : $sup_{2.2}(p_1, p_7, p_3, p_6)$:– $sup_{2.1}(p_1, p_2)$, $A(p_2, p_7, p_3, p_6)$.
r_5 : $sup_{2.3}(p_1, p_7, p_6, p_4)$:– $sup_{2.2}(p_1, p_7, p_3, p_6)$, $\mathsf{b}(p_3, p_4)$.
r_6 : $sup_{2.4}(p_1, p_7, p_4)$:– $sup_{2.3}(p_1, p_7, p_6, p_4)$, $\mathsf{c}(p_4, p_6)$.
r_7 : $B(p_1, p_8, p_4)$:– $sup_{2.4}(p_1, p_7, p_4)$, $\mathsf{d}(p_7, p_8)$.
r_8 : $sup_{3.1}(p_1, p_2)$:– $m\_B(p_1)$, $\mathsf{a}(p_1, p_2)$.
r_9 : $sup_{3.2}(p_1, p_4)$:– $sup_{3.1}(p_1, p_2)$, $\mathsf{b}(p_2, p_4)$.
r_{10} : $sup_{3.3}(p_1, p_4, p_6)$:– $sup_{3.2}(p_1, p_4)$, $\mathsf{c}(p_4, p_6)$.
r_{11} : $B(p_1, p_8, p_4)$:– $sup_{3.3}(p_1, p_4, p_6)$, $\mathsf{d}(p_6, p_8)$.
r_{12} : $m\_A(p_2)$:– $sup_{2.1}(p_1, p_2)$.
r_{13} : $m\_A(p_2)$:– $sup_{4.1}(p_1, p_2)$.
r_{14} : $sup_{4.1}(p_1, p_2)$:– $m\_A(p_1)$, $\mathsf{a}(p_1, p_2)$.
r_{15} : $sup_{4.2}(p_1, p_7, p_3, p_6)$:– $sup_{4.1}(p_1, p_2)$, $A(p_2, p_7, p_3, p_6)$.
r_{16} : $sup_{4.3}(p_1, p_7, p_6, p_4)$:– $sup_{4.2}(p_1, p_7, p_3, p_6)$, $\mathsf{b}(p_3, p_4)$.
r_{17} : $sup_{4.4}(p_1, p_7, p_4, p_5)$:– $sup_{4.3}(p_1, p_7, p_6, p_4)$, $\mathsf{c}(p_5, p_6)$.
r_{18} : $A(p_1, p_8, p_4, p_5)$:– $sup_{4.4}(p_1, p_7, p_4, p_5)$, $\mathsf{d}(p_7, p_8)$.
r_{19} : $sup_{5.1}(p_1, p_2)$:– $m\_A(p_1)$, $\mathsf{a}(p_1, p_2)$.
r_{20} : $sup_{5.2}(p_1, p_4)$:– $sup_{5.1}(p_1, p_2)$, $\mathsf{b}(p_2, p_4)$.
r_{21} : $sup_{5.3}(p_1, p_4, p_5, p_6)$:– $sup_{5.2}(p_1, p_4)$, $\mathsf{c}(p_5, p_6)$.
r_{22} : $A(p_1, p_8, p_4, p_5)$:– $sup_{5.3}(p_1, p_4, p_5, p_6)$, $\mathsf{d}(p_6, p_8)$.

The following version of chart parsing adds control structure to this deduction system:

1. (INIT) Initialize the chart to the empty set, the agenda to the singleton $\{m\_S(0)\}$, and n to 0.
2. Repeat the following steps:
   (a) Repeat the following steps until the agenda is exhausted:
      i. Remove a fact from the agenda, called the *trigger*.
      ii. Add the trigger to the chart.
      iii. Generate all facts that are immediate consequences of the trigger together with all facts in the chart, and add to the agenda those generated facts that are neither already in the chart nor in the agenda.
   (b) (SCAN) Remove the next fact from the input database and add it to the agenda, incrementing n. If there is no more fact in the input database, go to step 3.
3. If $S(0, n)$ is in the chart, accept; otherwise reject.

The following is the trace of the algorithm on input string aabbccdd:

1.	$m\_S(0)$	INIT	14.	$\mathsf{c}(4, 5)$	SCAN
2.	$m\_B(0)$	$r_1, 1$	15.	$sup_{5.3}(1, 3, 4, 5)$	$r_{21}, 12, 14$
3.	$\mathsf{a}(0, 1)$	SCAN	16.	$\mathsf{c}(6, 5)$	SCAN
4.	$sup_{2.1}(0, 1)$	$r_3, 2, 3$	17.	$sup_{5.3}(1, 3, 5, 6)$	$r_{21}, 12, 16$
5.	$sup_{3.1}(0, 1)$	$r_8, 2, 3$	18.	$\mathsf{d}(6, 7)$	SCAN
6.	$m\_A(1)$	$r_{12}, 4$	19.	$A(1, 7, 3, 5)$	$r_{22}, 17, 18$
7.	$\mathsf{a}(1, 2)$	SCAN	20.	$sup_{2.2}(0, 7, 3, 5)$	$r_4, 4, 19$
8.	$sup_{4.1}(1, 2)$	$r_{14}, 6, 7$	21.	$sup_{2.3}(0, 7, 5, 4)$	$r_5, 13, 20$
9.	$sup_{5.1}(1, 2)$	$r_{19}, 6, 7$	22.	$sup_{2.4}(0, 7, 4)$	$r_6, 14, 21$
10.	$m\_A(2)$	$r_{13}, 8$	23.	$\mathsf{d}(7, 8)$	SCAN
11.	$\mathsf{b}(2, 3)$	SCAN	24.	$B(0, 8, 4)$	$r_7, 22, 23$
12.	$sup_{5.2}(1, 3)$	$r_{20}, 9, 11$	25.	$S(0, 8)$	$r_2, 2, 24$
13.	$\mathsf{b}(3, 4)$	SCAN			

Note that unlike existing Earley-style parsing algorithms for TAGs, the present algorithm is an instantiation of a general schema that applies to parsing with more powerful grammar formalisms as well as to generation with Montague semantics.

8 Conclusion

Our reduction to Datalog brings sophisticated techniques for Datalog query evaluation to the problems

of parsing and generation, and establishes a tight bound on the computational complexity of recognition for a wide range of grammars. In particular, it shows that the use of higher-order λ-terms for semantic representation need not be avoided for the purpose of achieving computational tractability.

References

Aoto, Takahito. 1999. Uniqueness of normal proofs in implicational intuitionistic logic. *Journal of Logic, Language and Information* **8**, 217–242.

Aoto, Takahito and Hiroakira Ono. 1994. Uniqueness of normal proofs in $\{\rightarrow, \wedge\}$-fragment of NJ. Research Report IS-RR-94-0024F. School of Information Science, Japan Advanced Institute of Science and Technology.

Beeri, Catriel and Raghu Ramakrishnan. 1991. On the power of magic. *Journal of Logic Programming* **10**, 255–299.

Engelfriet, J. and E. M. Schmidt. 1977. IO and OI, part I. *The Journal of Computer and System Sciences* **15**, 328–353.

Engelfriet, Joost. 1986. The complexity of languages generated by attribute grammars. *SIAM Journal on Computing* **15**, 70–86.

Fisher, Michael J. 1968. *Grammars with Macro-Like Productions*. Ph.D. dissertation. Harvard University.

Gottlob, Georg, Nicola Lenoe, Francesco Scarcello. 2002. Computing LOGCFL certificates. *Theoretical Computer Science* **270**, 761–777.

de Groote, Philippe. 2001. Towards abstract categorial grammars. In *Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference*, pages 148–155.

de Groote, Philippe. 2002. Tree-adjoining grammars as abstract categorial grammars. In *Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6)*, pages 145–150. Universitá di Venezia.

de Groote, Philippe and Sylvain Pogodalla. 2004. On the expressive power of abstract categorial grammars: Representing context-free formalisms. *Journal of Logic, Language and Information* **13**, 421–438.

Hindley, J. Roger. 1997. *Basic Simple Type Theory*. Cambridge: Cambridge University Press.

Aravind K. Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In Grzegoz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages, Vol. 3*, pages 69–123. Berlin: Springer.

Kanazawa, Makoto and Ryo Yoshinaka. 2005. Lexicalization of second-order ACGs. NII Technical Report. NII-2005-012E. National Institute of Informatics, Tokyo.

Kanellakis, Paris C. 1988. Logic programming and parallel complexity. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 547–585. Los Altos, CA: Morgan Kaufmann.

Mints, Grigori. 2000. *A Short Introduction to Intuitionistic Logic*. New York: Kluwer Academic/Plenum Publishers.

Moore, Robert C. 2002. A complete, efficient sentence-realization algorithm for unification grammar. In *Proceedings, International Natural Language Generation Conference*, Harriman, New York, pages 41–48.

Salvati, Sylvain. 2005. *Problèmes de filtrage et problèmes d'analyse pour les grammaires catégorielles abstraites*. Doctoral dissertation, l'Institut National Polytechnique de Lorraine.

Salvati, Sylvain. 2007. Encoding second order string ACG with deterministic tree walking transducers. In Shuly Wintner, editor, *Proceedings of FG 2006: The 11th conference on Formal Grammar*, pages 143–156. FG Online Proceedings. Stanford, CA: CSLI Publications.

Seki, Hiroyuki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science* **88**, 191–229.

Shieber, Stuart M., Yves Schabes, and Fernando C. N. Pereira. 1995. Principles and implementations of deductive parsing. *Journal of Logic Programming* **24**, 3–36.

Sikkel, Klaas. 1997. *Parsing Schemata*. Berlin: Springer.

Sørensen, Morten Heine and Paweł Urzyczyn. 2006. *Lectures on the Curry-Howard Isomorphism*. Amsterdam: Elsevier.

Ullman, Jeffrey D. 1988. *Principles of Database and Knowledge-Base Systems. Volume I.* Rockville, MD.: Computer Science Press.

Ullman, Jeffrey D. 1989a. Bottom-up beats top-down for Datalog. In *Proceedings of the Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Philadelphia, pages 140–149.

Ullman, Jeffrey D. 1989b. *Principles of Database and Knowledge-Base Systems. Volume II: The New Technologies*. Rockville, MD.: Computer Science Press.

Ullman, Jeffrey D. and Allen Van Gelder. 1988. Parallel complexity of logical query programs. *Algorithmica* **3**, 5–42.

David J. Weir. 1988. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Ph.D. dissertation. University of Pennsylvania.

Optimizing Grammars for Minimum Dependency Length

Daniel Gildea
Computer Science Dept.
University of Rochester
Rochester, NY 14627

David Temperley
Eastman School of Music
University of Rochester
Rochester, NY 14604

Abstract

We examine the problem of choosing word order for a set of dependency trees so as to minimize total dependency length. We present an algorithm for computing the optimal layout of a single tree as well as a numerical method for optimizing a grammar of orderings over a set of dependency types. A grammar generated by minimizing dependency length in unordered trees from the Penn Treebank is found to agree surprisingly well with English word order, suggesting that dependency length minimization has influenced the evolution of English.

1 Introduction

Dependency approaches to language assume that every word in a sentence is the dependent of one other word (except for one word, which is the global head of the sentence), so that the words of a sentence form an acyclic directed graph. An important principle of language, supported by a wide range of evidence, is that there is preference for dependencies to be short. This has been offered as an explanation for numerous psycholinguistic phenomena, such as the greater processing difficulty of object relative clauses versus subject relative clauses (Gibson, 1998). Dependency length minimization is also a factor in ambiguity resolution: listeners prefer the interpretation with shorter dependencies. Statistical parsers make use of features that capture dependency length (e.g. an adjacency feature in Collins (1999), more explicit length features in McDonald et al. (2005) and Eisner and Smith (2005)) and thus learn to favor parses with shorter dependencies.

In this paper we attempt to measure the extent to which basic English word order chooses to minimize dependency length, as compared to average dependency lengths under other possible grammars. We first present a linear-time algorithm for finding the ordering of a single dependency tree with shortest total dependency length. Then, given that word order must also be determined by grammatical relations, we turn to the problem of specifying a grammar in terms of constraints over such relations. We wish to find the set of ordering constraints on dependency types that minimizes a corpus's total dependency length. Even assuming that dependency trees must be projective, this problem is NP-complete,[1] but we find that numerical optimization techniques work well in practice. We reorder unordered dependency trees extracted from corpora and compare the results to English in terms of both the resulting dependency length and the strings that are produced. The optimized order constraints show a high degree of similarity to English, suggesting that dependency length minimization has influenced the word order choices of basic English grammar.

2 The Dependency Length Principle

This idea that dependency length minimization may be a general principle in language has been discussed by many authors. One example concerns the

[1] English has crossing (non-projective) dependencies, but they are believed to be very infrequent. McDonald et al. (2005) report that even in Czech, commonly viewed as a non-projective language, fewer than 2% of dependencies violate the projectivity constraint.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 184–191,
Prague, Czech Republic, June 2007.

well-known principle that languages tend to be predominantly "head-first" (in which the head of each dependency is on the left) or "head-last" (where it is on the right). Frazier (1985) suggests that this might serve the function of keeping heads and dependents close together. In a situation where each word has exactly one dependent, it can be seen that a "head-first" arrangement achieves minimal dependency length, as each link has a length of one.

We will call a head-first dependency "right-branching" and a head-last dependency "left-branching"; a language in which most or all dependencies have the same branching direction is a "same-branching" language.

Another example of dependency length minimization concerns situations where a head has multiple dependents. In such cases, dependency length will be minimized if the shorter dependent is placed closer to the head. Hawkins (1994) has shown that this principle is reflected in grammatical rules across many languages. It is also reflected in situations of choice; for example, in cases where a verb is followed by a prepositional phrase and a direct object NP, the direct object NP will usually be placed first (closer to the verb) but if it is longer than the PP, it is often placed second.

While one might suppose that a "same-branching" language is optimal for dependency-length minimization, this is not in fact the case. If a word has several dependents, placing them all on the same side causes them to get in the way of each other, so that a more 'balanced" configuration – with some dependents on each side – has lower total dependency length. It is particularly desirable for one or more one-word dependent phrases to be "opposite-branching" (in relation to the prevailing branching direction of the language); opposite-branching of a long phrase tends to cause a long dependency from the head of the phrase to the external head.

Exactly this pattern has been observed by Dryer (1992) in natural languages. Dryer argues that, while most languages have a predominant branching direction, phrasal (multi-word) dependents tend to adhere to this prevailing direction much more consistently than one-word dependents, which frequently branch opposite to the prevailing direction of the language. English reflects this pattern quite strongly: While almost all phrasal dependents are right-branching (prepositional phrases, objects of prepositions and verbs, relative clauses, etc.), some 1-word categories are left-branching, notably determiners, noun modifiers, adverbs (sometimes), and attributive adjectives.

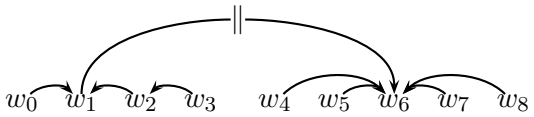


Figure 1: Separating a dependency link into two pieces at a subtree boundary.

This linguistic evidence strongly suggests that languages have been shaped by principles of dependency length minimization. One might wonder how close natural languages are to being optimal in this regard. To address this question, we extract unordered dependency graphs from English and consider different algorithms, which we call Dependency Linearization Algorithms (DLAs), for ordering the words; our goal is to find the algorithm that is optimal with regard to dependency length minimization. We begin with an "unlabeled" DLA, which simply minimizes dependency length without requiring consistent ordering of syntactic relations. We then consider the more realistic case of a "labeled" DLA, which is required to have syntactically consistent ordering.

Once we find the optimal DLA, two questions can be asked. First, how close is dependency length in English to that of this optimal DLA? Secondly, how similar is the optimal DLA to English in terms of the actual rules that arise?

3 The Optimal Unlabeled DLA

Finding linear arrangements of graphs that minimize total edge length is a classic problem, NP-complete for general graphs but with an $O(n^{1.6})$ algorithm for trees (Chung, 1984). However, the traditional problem description does not take into account the projectivity constraint of dependency grammar. This constraint simplifies the problem; in this section we show that a simple linear-time algorithm is guaranteed to find an optimal result.

A natural strategy would be to apply dynamic programming over the tree structure, observing that to-

tal dependency length of a linearization can be broken into the sum of links below any node w in the tree, and the sum of links outside the node, by which we mean all links not connected to dependents of the node. These two quantities interact only through the position of w relative to the rest of its descendants, meaning that we can use this position as our dynamic programming state, compute the optimal layout of each subtree given each position of the head within the subtree, and combine subtrees bottom-up to compute the optimal linearization for the entire sentence.

This can be further improved by observing that the total length of the outside links depends on the position of w only because it affects the length of the link connecting w to its parent. All other outside links either cross above all words under w, and depend only on the total size of w's subtree, or are entirely on one side of w's subtree. The link from w to its parent is divided into two pieces, whose lengths add up to the total length of the link, by slicing the link where it crosses the boundary from w's subtree to the rest of the sentence. In the example in Figure 1, the dependency from w_1 to w_6 has total length five, and is divided in to two components of length 2.5 at the boundary of w_1's subtree. The length of the piece over w's subtree depends on w's position within that subtree, while the other piece does not depend on the internal layout of w's subtree. Thus the total dependency length for the entire sentence can be divided into:

1. the length of all links within w's subtree plus the length of the first piece of w's link to its parent, i.e. the piece that is above descendants of w.

2. the length of the remaining piece of w's link to its parent plus the length of all links outside w.

where the second quantity can be optimized independently of the internal layout of w's subtree. While the link from w to its parent may point either to the right or left, the optimal layout for w's subtree given that w attaches to its left must be the mirror image of the optimal layout given that w attaches to its right. Thus, only one case need be considered, and the optimal layout for the entire sentence can be computed from the bottom up using just one dynamic programming state for each node in the tree.

We now go on to show that, in computing the ordering of the d_i children of a given node, not all $d_i!$ possibilities need be considered. In fact, one can simply order the children by adding them in increasing order of size, going from the head outwards, and alternating between adding to the left and right edges of the constituent.

The first part of this proof is the observation that, as we progress from the head outward, to either the left or the right, the head's child subtrees must be placed in increasing order of size. If any two adjacent children appear with the smaller one further from the head, we can swap the positions of these two children, reducing the total dependency length of the tree. No links crossing over the two children will change in length, and no links within either child will change. Thus only the length of the links from the two children will change, and as the link connecting the outside child now crosses over a shorter intermediate constituent, the total length will decrease.

Next, we show that the two longest children must appear on opposite sides of the head in the optimal linearization. To see this, consider the case where both child i (the longest child) and child $i-1$ (the second longest child) appear on the same side of the head. From the previous result, we know that $i-1$ and i must be the outermost children on their side. If there are no children on the other side of the head, the tree can be improved by moving either i or $i-1$ to the other side. If there is a child on the other side of the head, it must be smaller than both i and $i-1$, and the tree can be improved by swapping the position of the child from the other side and child $i-1$.

Given that the two largest children are outermost and on opposite sides of the head, we observe that the sum of the two links connecting these children to the head does not depend on the arrangement of the first $i-2$ children. Any rearrangement that decreases the length of the link to the left of the head must increase the length of the link to the right of the head by the same amount. Thus, the optimal layout of all i children can be found by placing the two largest children outermost and on opposite sides, the next two largest children next outermost and on op-

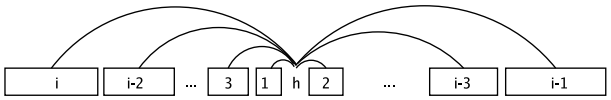


Figure 2: Placing dependents on alternating sides from inside out in order of increasing length.

DLA	Length
Optimal	33.7
Random	76.1
Observed	47.9

Table 1: Dependency lengths for unlabeled DLAs.

posite sides, and so on until only one or zero children are left. If there are an odd number of children, the side of the final (smallest) child makes no difference, because the other children are evenly balanced on the two sides so the last child will have the same dependency-lengthening effect whichever side it is on.

Our pairwise approach implies that there are many optimal linearizations, $2^{\lfloor i/2 \rfloor}$ in fact, but one simple and optimal approach is to alternate sides as in Figure 2, putting the smallest child next to the head, the next smallest next to the head on the opposite side, the next outside the first on the first side, and so on.

So far we have not considered the piece of the link from the head to its parent that is over the head's subtree. The argument above can be generalized by considering this link as a special child, longer than the longest real child. By making the special child the longest child, we will be guaranteed that it will be placed on the outside, as is necessary for a projective tree. As before, the special child and the longest real child must be placed outermost and on opposite sides, the next two longest children immediately within the first two, and so on.

Using the algorithm from the previous section, it is possible to efficiently compute the optimal dependency length from English sentences. We take sentences from the Wall Street Journal section of the Penn Treebank, extract the dependency trees using the head-word rules of Collins (1999), consider them to be unordered dependency trees, and linearize them to minimize dependency length. Automatically extracting dependencies from the Treebank can lead to some errors, in particular with complex compound nouns. Fortunately, compound nouns tend to occur at the leaves of the tree, and the head rules are reliable for the vast majority of structures.

Results in Table 1 show that observed dependency lengths in English are between the minimum achievable given the unordered dependencies and the length we would find given a random ordering, and are much closer to the minimum. This already suggests that minimizing dependency length has been a factor in the development of English. However, the optimal "language" to which English is being compared has little connection to linguistic reality. Essentially, this model represents a free word-order language: Head-modifier relations are oriented without regard to the grammatical relation between the two words. In fact, however, word order in English is relatively rigid, and a more realistic experiment would be to find the optimal algorithm that reflects consistent syntactic word order rules. We call this a "labeled" DLA, as opposed to the "unlabeled" DLA presented above.

4 Labeled DLAs

In this section, we consider linearization algorithms that assume fixed word order for a given grammatical relation, but choose the order such as to minimize dependency length over a large number of sentences. We represent grammatical relations simply by using the syntactic categories of the highest constituent headed by (maximal projection of) the two words in the dependency relation. Due to sparse data concerns, we removed all function tags such as TMP (temporal), LOC (locative), and CLR (closely related) from the treebank. We made an exception for the SBJ (subject) tag, as we thought it important to distinguish a verb's subject and object for the purposes of choosing word order. Looking at a head and its set of dependents, the complete ordering of all dependents can be modeled as a context-free grammar rule over a nonterminal alphabet of maximal projection categories. A fixed word-order language will have only one rule for each set of nonterminals appearing in the right-hand side.

Searching over all such DLAs would be exponentially expensive, but a simple approximation of the

DLA	Dep. len. / % correct order
random	76.1 / 40.5
extracted from optimal	61.6 / 55.4
weights from English	50.9 / 82.2
optimized weights	42.5 / 64.9

Table 2: Results for different methods of linearizing unordered trees from section 0 of the Wall Street Journal corpus. Each result is given as average dependency length in words, followed by the percentage of heads (with at least one dependent) having all dependents correctly ordered.

optimal labeled DLA can found using the following procedure:

1. Compute the optimal layout of all sentences in the corpus using the unlabeled DLA.

2. For each combination of a head type and a set of child types, count the occurrences of each ordering.

3. Take the most frequent ordering for each set as the order in the new DLA.

In the first step we used the alternating procedure from the previous section, with a modification for the fixed word-order scenario. In order to make the order of a subtree independent of the direction in which it attaches to its parent, dependents were placed in order of length on alternating sides of the head from the inside out, always starting with the shortest dependent immediately to the left of the head.

Results in Table 2 (first two lines) show that a DLA using rules extracted from the optimal layout matches English significantly better than a random DLA, indicating that dependency length can be used as a general principle to predict word order.

4.1 An Optimized Labeled DLA

While the DLA presented above is a good deal better than random (in terms of minimizing dependency length), there is no reason to suppose that it is optimal. In this section we address the issue of finding the optimal labeled DLA.

If we model a DLA as a set of context-free grammar rules over dependency types, specifying a fixed ordering for any set of dependency types attaching to a given head, the space of DLAs is enormous, and the problem of finding the optimal DLA is a difficult one. One way to break the problem down is to model the DLA as a set of weights for each type of dependency relation. Under this model the word order is determined by placing all dependents of a word in order of increasing weight from left to right. This reduces the number of parameters of the model to T, if there are T dependency types, from T^k if a word may have up to k dependents. It also allows us to naturally capture statements such as "a noun phrase consists of a determiner, then (possibly) some adjectives, the head noun, and then (possibly) some prepositional phrases", by, for example, setting the weight for NP→DT to -2, NP→JJ to -1, and NP→PP to 1. We assume the head itself has a weight of zero, meaning negatively weighted dependents appear to the head's left, and positively weighted dependents to the head's right.

4.1.1 A DLA Extracted from English

As a test of whether this model is adequate to represent English word order, we extracted weights for the Wall Street Journal corpus, used them to reorder the same set of sentences, and tested how often words with at least one dependent were assigned the correct order. We extracted the weights by assigning, for each dependency relation in the corpus, an integer according to its position relative to the head, -1 for the first dependent to the left, -2 for the second to the left, and so on. We averaged these numbers across all occurrences of each dependency type. The dependency types consisted of the syntactic categories of the maximal projections of the two words in the dependency relation.

Reconstructing the word order of each sentence from this weighted DLA, we find that 82% of all words with at least one dependent have all dependents ordered correctly (third line of Table 2). This is significantly higher than the heuristic discussed in the previous section, and probably as good as can be expected from such a simple model, particularly in light of the fact that there is some choice in the word order for most sentences (among adjuncts for example) and that this model does not take the lengths of

the individual constituents into account at all.

We now wish to find the set of weights that minimize the dependency length of the corpus. While the size of the search space is still too large to search exhaustively, numerical optimization techniques can be applied to find an approximate solution.

4.1.2 NP-Completeness

The problem of finding the optimum weighted DLA for a set of input trees can be shown to be NP-complete by reducing from the problem of finding a graph's minimum Feedback Arc Set, one of the 21 classic problems of Karp (1972). The input to the Feedback Arc Set problem is a directed graph, for which we wish to find an ordering of vertices such that the smallest number of edges point from later to earlier vertices in the ordering. Given an instance of this problem, we can create a set of dependency trees such that each feedback arc in the original graph causes total dependency length to increase by one, if we identify each dependency type with a vertex in the original problem, and choose weights for the dependency types according to the vertex order.[2]

4.1.3 Local Search

Our search procedure is to optimize one weight at a time, holding all others fixed, and iterating through the set of weights to be set. The objective function describing the total dependency length of the corpus is piecewise constant, as the dependency length will not change until one weight crosses another, causing two dependents to reverse order, at which point the total length will discontinuously jump. Non-differentiability implies that methods based on gradient ascent will not apply. This setting is reminiscent of the problem of optimizing feature weights for reranking of candidate machine translation outputs, and we employ an optimization technique similar to that used by Och (2003) for machine translation. Because the objective function only changes at points where one weight crosses another's value, the set of segments of weight values with different values of the objective function can be exhaustively enumerated. In fact, the only significant points are the values of other weights for dependency types which occur in the corpus attached to the same head as the dependency being optimized. We build a table of interacting dependencies as a preprocessing step on the data, and then when optimizing a weight, consider the sequence of values between consecutive interacting weights. When computing the total corpus dependency length at a new weight value, we can further speed up computation by reordering only those sentences in which a dependency type is used, by building an index of where dependency types occur as another preprocessing step.

[2] We omit details due to space.

	Test Data	
Training Data	WSJ	Swbd
WSJ	42.5 / 64.9	12.5 / 63.6
Swbd	43.9 / 59.8	12.2 / 58.7

Table 3: Domain effects on dependency length minimization: each result is formatted as in Table 2.

This optimization process is not guaranteed to find the global maximum (for this reason we call the resulting DLA "optimized" rather than "optimal"). The procedure is guaranteed to converge simply from the fact that there are a finite number of objective function values, and the objective function must increase at each step at which weights are adjusted.

We ran this optimization procedure on section 2 through 21 of the Wall Street Journal portion of the Penn Treebank, initializing all weights to random numbers between zero and one. This initialization makes all phrases head-initial to begin with, and has the effect of imposing a directional bias on the resulting grammar. When optimization converges, we obtain a set of weights which achieves an average dependency length of 40.4 on the training data, and 42.5 on held-out data from section 0 (fourth line of Table 2). While the procedure is unsupervised with respect to the English word order (other than the head-initial bias), it is supervised with respect to dependency length minimization; for this reason we report all subsequent results on held-out data. While random initializations lead to an initial average dependency length varying from 60 to 73 with an average of 66 over ten runs, all runs were within $\pm.5$ of one another upon convergence. When the order of words' dependents was compared to the real word order on held-out data, we find that 64.9% of words

Training Sents	Dep. len. / % correct order
100	13.70 / 54.38
500	12.81 / 57.75
1000	12.59 / 58.01
5000	12.34 / 55.33
10000	12.27 / 55.92
50000	12.17 / 58.73

Table 4: Average dependency length and rule accuracy as a function of training data size, on Switchboard data.

with at least one dependent have the correct order.

4.2 Domain Variation

Written and spoken language differ significantly in their structure, and one of the most striking differences is the much greater average sentence length of formal written language. The Wall Street Journal is not representative of typical language use. Language was not written until relatively recently in its development, and the Wall Street Journal in particular represents a formal style with much longer sentences than are used in conversational speech. The change in the lengths of sentences and their constituents could make the optimized DLA in terms of dependency length very different for the two genres.

In order to test this effect, we performed experiments using both the Wall Street Journal (written) and Switchboard (conversational speech) portions of the Penn Treebank, and compared results with different training and test data. For Switchboard, we used the first 50,000 sentences of sections 2 and 3 as the training data, and all of section 4 as the test data.

We find relatively little difference in dependency length as we vary training data between written and spoken English, as shown in Table 3. For the accuracy of the resulting word order, however, training on Wall Street Journal outperforms Switchboard even when testing on Switchboard, perhaps because the longer sentences in WSJ provide more information for the optimization procedure to work with.

4.3 Learning Curve

How many sentences are necessary to learn a good set of dependency weights? Table 4 shows results for Switchboard as we increase the number of sentences provided as input to the weight optimization procedure. While the average dependency length on held-out test data slowly decreases with more data, the percentage of correctly ordered dependents is less well-behaved. It turns out that even 100 sentences are enough to learn a DLA that is nearly as good as one derived from a much larger dataset.

Label	Interpretation	Weight
S→NP	verb - object NP	0.037
S→NP-SBJ	verb - subject NP	-0.022
S→PP	verb - PP	0.193
NP→DT	object noun - determiner	-0.070
NP-SBJ→DT	subject noun - determiner	-0.052
NP→PP	obj noun - PP	0.625
NP-SBJ→PP	subj noun - PP	0.254
NP→SBAR	obj noun - rel. clause	0.858
NP-SBJ→SBAR	subject noun - rel. clause	-0.110
NP→JJ	obj noun - adjective	0.198
NP-SBJ→JJ	subj noun - adjective	-0.052

Table 5: Sample weights from optimized DLA. Negatively weighted dependents appear to the left of their head.

4.4 Comparing the Optimized DLA to English

We have seen that the optimized DLA matches English text much better than a random DLA and that it achieves only a slightly lower dependency length than English. It is also of interest to compare the optimized DLA to English in more detail. First we examine the DLA's tendency towards "opposite-branching 1-word phrases". English reflects this principle to a striking degree: on the WSJ test set, 79.4 percent of left-branching phrases are 1-word, compared to only 19.4 percent of right-branching phrases. The optimized DLA also reflects this pattern, though somewhat less strongly: 75.5 percent of left-branching phrases are 1-word, versus 36.7 percent of right-branching phrases.

We can also compare the optimized DLA to English with regard to specific rules. As explained earlier, the optimal DLA's rules are expressed in the form of weights assigned to each relation, with positive weights indicating right-branching placement. Table 5 shows some important rules. The middle column shows the syntactic situation in which the relation normally occurs. We see, first of all, that object NPs are to the right of the verb and subject NPs are to the left, just like in English. PPs are also the right of verbs; the fact that the weight is greater than for NPs indicates that they are placed further to the right, as they normally are in English. Turning

to the internal structure of noun phrases, we see that determiners are to the left of both object and subject nouns; PPs are to the right of both object and subject nouns. We also find some differences with English, however. Clause modifiers of nouns (these are mostly relative clauses) are to the right of object nouns, as in English, but to the left of subject nouns; adjectives are to the left of subject nouns, as in English, but to the right of object nouns. Of course, these differences partly arise from the fact that we treat NP and NP-SBJ as distinct whereas English does not (with regard to their internal structure).

5 Conclusion

In this paper we have presented a dependency linearization algorithm which is optimized for minimizing dependency length, while still maintaining consistent positioning for each grammatical relation. The fact that English is so much lower than the random DLAs in dependency length gives suggests that dependency length minimization is an important general preference in language. The output of the optimized DLA also proves to be much more similar to English than a random DLA in word order. An informal comparison of some important rules between English and the optimal DLA reveals a number of striking similarities, though also some differences.

The fact that the optimized DLA's ordering matches English on only 65% of words shows, not surprisingly, that English word order is determined by other factors in addition to dependency length minimization. In some cases, ordering choices in English are underdetermined by syntactic rules. For example, a manner adverb may be placed either before the verb or after ("He ran quickly / he quickly ran"). Here the optimized DLA requires a consistent ordering while English does not. One might suppose that such syntactic choices in English are guided at least partly by dependency length minimization, and indeed there is evidence for this; for example, people tend to put the shorter of two PPs closer to the verb (Hawkins, 1994). But there are also other factors involved – for example, the tendency to put "given" discourse elements before "new" ones, which has been shown to play a role independent of length (Arnold et al., 2000).

In other cases, the optimized DLA allows more fine-grained choices than English. For example, the optimized DLA treats NP and NP-SBJ as different; this allows it to have different syntactic rules for the two cases – a possibility that it sometimes exploits, as seen above. No doubt this partly explains why the optimized DLA achieves lower dependency length than English.

Acknowledgments This work was supported by NSF grants IIS-0546554 and IIS-0325646.

References

J. E. Arnold, T. Wasow, T. Losongco, and R. Ginstrom. 2000. Heaviness vs. newness: the effects of structural complexity and discourse status on constituent ordering. *Language*, 76:28–55.

F. R. K. Chung. 1984. On optimal linear arrangements of trees. *Computers and Mathematics with Applications*, 10:43–60.

Michael John Collins. 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia.

Matthew Dryer. 1992. The Greenbergian word order correlations. *Language*, 68:81–138.

Jason Eisner and Noah A. Smith. 2005. Parsing with soft and hard constraints on dependency length. In *Proceedings of the International Workshop on Parsing Technologies (IWPT)*, pages 30–41.

Lyn Frazier. 1985. Syntactic complexity. In D. Dowty, L. Karttunen, and A. Zwicky, editors, *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives*, pages 129–189. Cambridge University Press, Cambridge.

Edward Gibson. 1998. Linguistic complexity: Locality of syntactic dependencies. *Cognition*, 68:1–76.

John Hawkins. 1994. *A Performance Theory of Order and Constituency*. Cambridge University Press, Cambridge, UK.

Richard M. Karp. 1972. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103.

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

Franz Josef Och. 2003. Minimum error rate training for statistical machine translation. In *Proceedings of ACL-03*.

Generalizing Semantic Role Annotations Across Syntactically Similar Verbs

Andrew S. Gordon
Institute for Creative Technologies
University of Southern California
Marina del Rey, CA 90292 USA
gordon@ict.usc.edu

Reid Swanson
Institute for Creative Technologies
University of Southern California
Marina del Rey, CA 90292 USA
swansonr@ict.usc.edu

Abstract

Large corpora of parsed sentences with semantic role labels (e.g. PropBank) provide training data for use in the creation of high-performance automatic semantic role labeling systems. Despite the size of these corpora, individual verbs (or rolesets) often have only a handful of instances in these corpora, and only a fraction of English verbs have even a single annotation. In this paper, we describe an approach for dealing with this sparse data problem, enabling accurate semantic role labeling for novel verbs (rolesets) with only a single training example. Our approach involves the identification of syntactically similar verbs found in PropBank, the alignment of arguments in their corresponding rolesets, and the use of their corresponding annotations in PropBank as surrogate training data.

1 Generalizing Semantic Role Annotations

A recent release of the PropBank (Palmer et al., 2005) corpus of semantic role annotations of Treebank parses contained 112,917 labeled instances of 4,250 rolesets corresponding to 3,257 verbs, as illustrated by this example for the verb *buy*.

[$_{arg0}$ Chuck] [$_{buy.01}$ bought] [$_{arg1}$ a car] [$_{arg2}$ from Jerry] [$_{arg3}$ for $1000].

Annotations similar to these have been used to create automated semantic role labeling systems (Pradhan et al., 2005; Moschitti et al., 2006) for use in natural language processing applications that require only shallow semantic parsing. As with all machine-learning approaches, the performance of these systems is heavily dependent on the availability of adequate amounts of training data. However, the number of annotated instances in PropBank varies greatly from verb to verb; there are 617 annotations for the *want* roleset, only 7 for *desire*, and 0 for any sense of the verb *yearn*. Do we need to keep annotating larger and larger corpora in order to generate accurate semantic labeling systems for verbs like *yearn*?

A better approach may be to generalize the data that exists already to handle novel verbs. It is reasonable to suppose that there must be a number of verbs within the PropBank corpus that behave nearly exactly like *yearn* in the way that they relate to their constituent arguments. Rather than annotating new sentences that contain the verb *yearn*, we could simply find these similar verbs and use their annotations as surrogate training data.

This paper describes an approach to generalizing semantic role annotations across different verbs, involving two distinct steps. The first step is to order all of the verbs with semantic role annotations according to their syntactic similarity to the target verb, followed by the second step of aligning argument labels between different rolesets. To evaluate this approach we developed a simple automated semantic role labeling algorithm based on the frequency of parse-tree paths, and then compared its performance when using real and surrogate training data from PropBank.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 192–199,
Prague, Czech Republic, June 2007.

2 Parse Tree Paths

A key concept in understanding our approach to both automated semantic role annotation and generalization is the notion of a *parse tree path*. Parse tree paths were used for semantic role labeling by Gildea and Jurafsky (2002) as descriptive features of the syntactic relationship between predicates and their arguments in the parse tree of a sentence. Predicates are typically assumed to be specific target words (verbs), and arguments are assumed to be spans of words in the sentence that are dominated by nodes in the parse tree. A parse tree path can be described as a sequence of transitions up from the target word then down to the node that dominates the argument span (e.g. Figure 1).

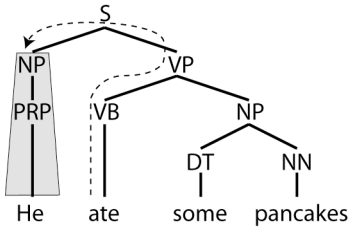


Figure 1: An example parse tree path from the predicate *ate* to the argument NP *He*, represented as ↑VB↑VP↑S↓NP

Parse tree paths are particularly interesting for automated semantic role labeling because they generalize well across syntactically similar sentences. For example, the parse tree path in Figure 1 would still correctly identify the "eater" argument in the given sentence if the personal pronoun "he" were swapped with a markedly different noun phrase, e.g. "the attendees of the annual holiday breakfast."

3 A Simple Semantic Role Labeler

To explore issues surrounding the generalization of semantic role annotations across verbs, we began by authoring a simple automated semantic role labeling algorithm that assigns labels according to the frequency of the parse tree paths seen in training data. To construct a labeler for a specific roleset, training data consisting of parsed sentences with role-labeled parse tree constituents are analyzed to identify all of the parse tree paths between predicates and arguments, which are then tabulated and sorted by frequency. For example, Table 1 lists the 10 most frequent pairs of arguments and parse tree paths for the *want.01* roleset in a recent release of PropBank.

Count	Argument	Parse tree path
189	ARG0	↑VBP↑VP↑S↓NP
159	ARG1	↑VBP↑VP↓S
125	ARG0	↑VBZ↑VP↑S↓NP
110	ARG1	↑VBZ↑VP↓S
102	ARG0	↑VB↑VP↑VP↑S↓NP
98	ARG1	↑VB↑VP↓S
96	ARG0	↑VBD↑VP↑S↓NP
79	ARGM	↑VB↑VP↑VP↓RB
76	ARG1	↑VBD↑VP↓S
43	ARG1	↑VBP↑VP↓NP

Table 1. Top 10 most frequent parse tree paths for arguments of the PropBank want.01 roleset, based on 617 annotations

To automatically assign role labels to an unlabeled parse tree, each entry in the table is considered in order of highest frequency. Beginning from the target word in the sentence (e.g. *wants*) a check is made to determine if the entry includes a possible parse tree path in the parse tree of the sentence. If so, then the constituent is assigned the role label of the entry, and all subsequent entries in the table that have the same argument label or lead to subconstituents of the labeled node are invalidated. Only subsequent entries that assign core arguments of the roleset (e.g. ARG0, ARG1) are invalidated, allowing for multiple assignments of non-core labels (e.g. ARGM) to a test sentence. In cases where the path leads to more than one node in a sentence, the leftmost path is selected. This process then continues down the list of valid table entries, assigning additional labels to unlabeled parse tree constituents, until the end of the table is reached.

This approach also offers a simple means of dealing with multiple-constituent arguments, which occasionally appear in PropBank data. In these cases, the data is listed as unique entries in the frequency table, where each of the parse tree paths to the multiple constituents are listed as a set. The labeling algorithm will assign the argument of the entry only if all parse tree paths in the set are present in the sentence.

The expected performance of this approach to semantic role labeling was evaluated using the PropBank data using a leave-one-out cross-validation experimental design. Precision and recall scores were calculated for each of the 3,086

rolesets with at least two annotations. Figure 2 graphs the average precision, recall, and F-score for rolesets according to the number of training examples of the roleset in the PropBank corpus. An additional curve in Figure 2 plots the percentage of these PropBank rolesets that have the given amount of training data or more. For example, F-scores above 0.7 are first reached with 62 training examples, but only 8% of PropBank rolesets have this much training data available.

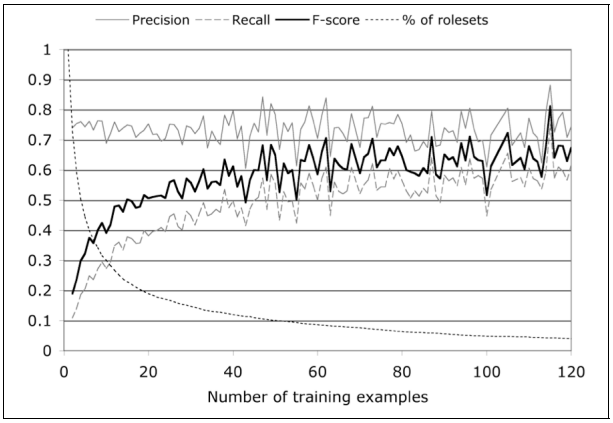


Figure 2. Performance of our semantic role labeling approach on PropBank rolesets

4 Identifying Syntactically Similar Verbs

A key part of generalizing semantic role annotations is to calculate the syntactic similarity between verbs. The expectation here is that verbs that appear in syntactically similar contexts are going to behave similarly in the way that they relate to their arguments. In this section we describe a fully automated approach to calculating the syntactic similarity between verbs.

Our approach is strictly empirical; the similarity of verbs is determined by examining the syntactic contexts in which they appear in a large text corpus. Our approach is analogous to previous work in extracting collocations from large text corpora using syntactic information (Lin, 1998). In our work, we utilized the GigaWord corpus of English newswire text (Linguistic Data Consortium, 2003), consisting of nearly 12 gigabytes of textual data. To prepare this corpus for analysis, we extracted the body text from each of the 4.1 million entries in the corpus and applied a maximum-entropy algorithm to identify sentence boundaries (Reynar and Ratnaparkhi, 1997).

Next we executed a four-step analysis process for each of the 3,257 verbs in the PropBank corpus. In the first step, we identified each of the sentences in the prepared GigaWord corpus that contained any inflection of the given verb. To automatically identify all verb inflections, we utilized the English DELA electronic dictionary (Courtois, 2004), which contained all but 21 of the PropBank verbs (for which we provided the inflections ourselves), with old-English verb inflections removed. We extracted GigaWord sentences containing these inflections by using the GNU *grep* program and a template regular expression for each inflection list. The results of these searches were collected in 3,257 files (one for each verb). The largest of these files was for inflections of the verb *say* (15.9 million sentences), and the smallest was for the verb *namedrop* (4 sentences).

The second step was to automatically generate syntactic parse trees for the GigaWord sentences found for each verb. It was our original intention to parse all of the found sentences, but we found that the slow speed of contemporary syntactic parsers made this impractical. Instead, we focused our efforts on the first 100 sentences found for each of the 3,257 verbs with 100 or fewer tokens: a total of 324,461 sentences (average of 99.6 per verb). For this task we utilized the August 2005 release of the Charniak parser with the default speed/accuracy settings (Charniak, 2000), which required roughly 360 hours of processor time on a 2.5 GHz PowerPC G5.

The third step was to characterize the syntactic context of the verbs based on where they appeared within the parse trees. For this purpose, we utilized parse tree paths as a means of converting tree structures into a flat, feature-vector representation. For each sentence, we identified all *possible* parse tree paths that begin from the verb inflection and terminate at a constituent that does not include the verb inflection. For example, the syntactic context of the verb in Figure 1 can be described by the following five parse tree paths:

1. ↑VB↑VP↑S↓NP
2. ↑VB↑VP↑S↓NP↓PRP
3. ↑VB↑VP↓NP
4. ↑VB↑VP↓NP↓DT
5. ↑VB↑VP↓NP↓NN

Possible parse tree paths were identified for every parsed sentence for a given verb, and the frequencies of each unique path were tabulated

into a feature vector representation. Parse tree paths where the first node was not a Treebank part-of-speech tag for a verb were discarded, effectively filtering the non-verb homonyms of the set of inflections. The resulting feature vectors were normalized by dividing the values of each feature by the number of verb instances used to generate the parse tree paths; the value of each feature indicates the proportion of observed inflections in which the parse tree path is possible. As a representative example, 95 verb forms of *abandon* were found in the first 100 GigaWord sentences containing any inflection of this verb. For this verb, 4,472 possible parse tree paths were tabulated into 3,145 unique features, 2501 of which occurred only once.

The fourth step was to compute the distance between a given verb and each of the 3,257 feature vector representations describing the syntactic context of PropBank verbs. We computed and compared the performance of a wide variety of possible vector-based distance metrics, including Euclidean, Manhattan, and Chi-square (with un-normalized frequency counts), but found that the ubiquitous cosine measure was least sensitive to variations in sample size between verbs. To facilitate a comparative performance evaluation (section 6), pairwise cosine distance measures were calculated between each pair of PropBank verbs and sorted into individual files, producing 3,257 lists of 3,257 verbs ordered by similarity.

Table 2 lists the 25 most syntactically similar pairs of verbs among all PropBank verbs. There are a number of notable observations in this list. First is the extremely high similarity between *bind* and *bound*. This is partly due to the fact that they share an inflection (*bound* is the irregular past tense form of *bind*), so the first 100 instances of GigaWord sentences for each verb overlap significantly, resulting in overlapping feature vector representations. Although this problem appears to be restricted to this one pair of verbs, it could be avoided in the future by using the part-of-speech tag in the parse tree to help distinguish between verb lemmas.

A second observation of Table 2 is that several verbs appear multiple times in this list, yielding sets of verbs that all have high syntactic similarity. Three of these sets account for 19 of the verbs in this list:

1. plunge, tumble, dive, jump, fall, fell, dip
2. assail, chide, lambaste
3. buffet, embroil, lock, superimpose, whipsaw, pluck, whisk, mar, ensconce

The appearance of these sets suggests that our method of computing syntactic similarity could be used to identify distinct clusters of verbs that behave in very similar ways. In future work, it would be particularly interesting to compare empirically-derived verb clusters to verb classes derived from theoretical considerations (Levin, 1993), and to the automated verb classification techniques that use these classes (Joanis and Stevenson, 2003).

A third observation of Table 2 is that the verb pairs with the highest syntactic similarity are often synonyms, e.g. the cluster of *assail*, *chide*, and *lambaste*. As a striking example, the 14 most syntactically similar verbs to *believe* (in order) are *think*, *guess*, *hope*, *feel*, *wonder*, *theorize*, *fear*, *reckon*, *contend*, *suppose*, *understand*, *know*, *doubt*, and *suggest* – all mental action verbs. This observation further supports the distributional hypothesis of word similarity and corresponding technologies for identifying synonyms by similarity of lexical-syntactic context (Lin, 1998).

Verb pairs (instances)		*Cosine*
bind (83)	bound (95)	0.950
plunge (94)	tumble (87)	0.888
dive (36)	plunge (94)	0.867
dive (36)	tumble (87)	0.866
jump (79)	tumble (87)	0.865
fall (84)	fell (102)	0.859
intersperse (99)	perch (81)	0.859
assail (100)	chide (98)	0.859
dip (81)	fell (102)	0.858
buffet (72)	embroil (100)	0.856
embroil (100)	lock (73)	0.856
embroil (100)	superimpose (100)	0.856
fell (102)	jump (79)	0.855
fell (102)	tumble (87)	0.855
embroil (100)	whipsaw (63)	0.850
pluck (100)	whisk (99)	0.849
acquit (100)	hospitalize (99)	0.849
disincline (70)	obligate (94)	0.848
jump (79)	plunge (94)	0.848
dive (36)	jump (79)	0.847
assail (100)	lambaste (100)	0.847
festoon (98)	strew (100)	0.846
mar (78)	whipsaw (63)	0.846
pluck (100)	whipsaw (63)	0.846
ensconce (101)	whipsaw (63)	0.845

Table 2. Top 25 most syntactically similar pairs of the 3257 verbs in PropBank. Each verb is listed with the number of inflection instances used to calculate the cosine measurement.

5 Aligning Arguments Across Rolesets

The second key aspect of our approach to generalizing annotations is to make mappings between the argument roles of the novel target verb and the roles used for a given roleset in the PropBank corpus. For example, if we'd like to apply the training data for a roleset of the verb *desire* in PropBank to a novel roleset for the verb *yearn*, we need to know that the *desirer* corresponds to the *yearner*, the *desired* to the *yearned-for*, etc. In this section, we describe an approach to argument alignment that involves the application of the semantic role labeling approach described in section 3 to a single training example for the target verb.

To simplify the process of aligning argument labels across rolesets, we make a number of assumptions. First, we only consider cases where two rolesets have exactly the same number of arguments. The version of the PropBank corpus that we used in this research contained 4250 rolesets, each with 6 or fewer roles (typically two or three). Accordingly, when attempting to apply PropBank data to a novel roleset with a given argument count (e.g. two), we only consider the subset of PropBank data that labels rolesets with exactly the same count.

Second, our approach requires at least one fully-annotated training example for the target roleset. A fully-annotated sentence is one that contains a labeled constituent in its parse tree for each role in the roleset. As an illustration, the example sentence in section 1 (for the roleset *buy.01*) would not be considered a fully-annotated training example, as only four of the five arguments of the PropBank *buy.01* roleset are present in the sentence (it is missing a *benefactor*, as in "Chuck bought *his mother* a car from Jerry for $1000").

In both of these simplifying requirements, we ignore role labels that may be assigned to a sentence but that are not defined as part of the roleset, specifically the *ARGM* labels used in PropBank to label standard proposition modifiers (e.g. location, time, manner).

Our approach begins with a list of verbs ordered by their calculated syntactic similarity to the target verb, as described in section 4 of this paper. We subsequently apply two steps that transform this list into an ordered set of rolesets that can be aligned with the roles used in one or more fully-annotated training examples of the target verb. In describing these two steps, we use *instigate* as an example target verb. Instigate already appears in the PropBank corpus as a two-argument roleset, but it has only a single training example:

> [$_{arg0}$ The Mahatma, or "great souled one,"] [$_{instigate.01}$ instigated] [$_{arg1}$ several campaigns of passive resistance against the British government in India].

The syntactic similarity of *instigate* to all PropBank verbs was calculated in the manner described in the previous section. This resulting list of 3,180 entries begins with the following fourteen verbs: *orchestrate*, *misrepresent*, *summarize*, *wreak*, *rub*, *chase*, *refuse*, *embezzle*, *harass*, *spew*, *thrash*, *unearth*, *snub*, and *erect*.

The first step is to replace each of the verbs in the ordered list with corresponding rolesets from PropBank that have the same number of roles as the target verb. As an example, our target roleset for the verb *instigate* has two arguments, so each verb in the ordered list is replaced with the set of corresponding rolesets that also have two arguments, or removed if no two-argument rolesets exist for the verb in the PropBank corpus. The ordered list of verbs for *instigate* is transformed into an ordered list of 2,115 rolesets with two arguments, beginning with the following five entries: *orchestrate.01*, *chase.01*, *unearth.01*, *snub.01*, and *erect.01*.

The second step is to identify the alignments between the arguments of the target roleset and each of the rolesets in the ordered list. Beginning with the first roleset on the list (e.g. *orchestrate.01*), we build a semantic role labeler (as described in section 3) using its available training annotations from the PropPank corpus. We then apply this labeler to the single, fully-annotated example sentence for the target verb, treating it as if it were a test example of the same roleset. We then check to see if any of the core (numbered) role labels *overlap* with the annotations that are provided. In cases where an annotated constituent of the target test sentence is assigned a label from the source roleset, then the roleset mappings are noted along with the entry in the ordered list. If no mappings are found, the roleset is removed from the ordered list.

For example, the roleset for *orchestrate.01* contains two arguments (*ARG0* and *ARG1*) that correspond to the "conductor, manager" and the "things

being coordinated or managed". This roleset is used for only three sentence annotations in the PropBank corpus. Using these annotations as training data, we build a semantic role labeler for this roleset and apply it to the annotated sentence for *instigate.01*, treating it as if it were a test sentence for the roleset *orchestrate.01*. The labeler assigns the *orchestrate.01* label *ARG1* to the same constituent labeled *ARG1* in the test sentence, but fails to assign a label to the other argument constituent in the test sentence. Therefore, a single mapping is recorded in the ordered list of rolesets, namely that *ARG1* of *orchestrate.01* can be mapped to *ARG1* of *instigate.01*.

After all of the rolesets are considered, we are left with a filtered list of rolesets with their argument mappings, ordered by their syntactic similarity to the target verb. For the roleset instigate.01, this list consists of 789 entries, beginning with the following 5 mappings.

1. *orchestrate.01*, 1:1
2. *chase.01*, 0:0, 1:1
3. *unearth.01*, 0:0, 1:1
4. *snub.01*, 1:1
5. *erect.01*, 0:0, 1:1

Given this list, arbitrary amounts of PropBank annotations can be used as surrogate training data for the *instigate.01* roleset, beginning at the top of the list. To utilize surrogate training data in our semantic role labeling approach (Section 3), we combine parse tree path information for a selected portion of surrogate training data into a single list sorted by frequency, and apply these files to test sentences as normal.

Although we use an existing PropBank roleset (*instigate.01*) as an example in this section, this approach will work for any novel roleset where one fully-annotated training example is available. For example, arbitrary amounts of surrogate PropBank data can be found for the novel verb *yearn* by 1) searching for sentences with the verb *yearn* in the GigaWord corpus, 2) calculating the syntactic similarity between *yearn* and all PropBank verbs as described in Section 4, 3) aligning the arguments in a single fully-annotated example of *yearn* with ProbBank rolesets with the same number of arguments using the method described in this section, and 4) selecting arbitrary amounts of PropBank annotations to use as surrogate training data, starting from the top of the resulting list.

6 Evaluation

We conducted a large-scale evaluation to determine the performance of our semantic role labeling algorithm when using variable amounts of surrogate training data, and compared these results to the performance that could be obtained using various amounts of real training data (as described in section 3). Our hypothesis was that learning-curves for surrogate-trained labelers would be somewhat less steep, but that the availability of large-amounts of surrogate training data would more than make up for the gap.

To test this hypothesis, we conducted an evaluation using the PropBank corpus as our testing data as well as our source for surrogate training data. As described in section 5, our approach requires the availability of at least one fully-annotated sentence for a given roleset. Only 28.5% of the PropBank annotations assign labels for each of the numbered arguments in their given roleset, and only 2,858 of the 4,250 rolesets used in PropBank annotations (66.5%) have at least one fully-annotated sentence. Of these, 2,807 rolesets were for verbs that appeared at least once in our analysis of the GigaWord corpus (Section 4). Accordingly, we evaluated our approach using the annotations for this set of 2,807 rolesets as test data. For each of these rolesets, various amounts of surrogate training data were gathered from all 4,250 rolesets represented in PropBank, leaving out the data for whichever roleset was being tested.

For each of the target 2,807 rolesets, we generated a list of semantic role mappings ordered by syntactic similarity, using the methods described in sections 4 and 5. In aligning arguments, only a single training example from the target roleset was used, namely the first annotation within the PropBank corpus where all of the rolesets arguments were assigned. Our approach failed to identify any argument mappings for 41 of the target rolesets, leaving them without any surrogate training data to utilize. Of the remaining 2,766 rolesets, the number of mapped rolesets for a given target ranged from 1,041 to 1 (mean = 608, stdev = 297).

For each of the 2,766 target rolesets with alignable roles, we gathered increasingly larger amounts of surrogate training data by descending the ordered list of mappings translating the PropBank data for each entry according to its argument mappings. Then each of these incrementally larger sets

of training data was then used to build a semantic role labeler as described in section 3. The performance of each of the resulting labelers was then evaluated by applying it to all of the test data available for target roleset in PropBank, using the same scoring methods described in section 3. The performance scores for each labeler were recorded along with the total number of surrogate training examples used to build the labeler.

Figure 3 presents the performance result of our semantic role labeling approach using various amounts of surrogate training data. Along with precision, recall, and F-score data, Figure 3 also graphs the percentage of PropBank rolesets for which a given amount of training data had been identified using our approach, of the 2,858 rolesets with at least one fully-annotated training example. For instance, with 120 surrogate annotations our system achieves an F-score above 0.5, and we identified this much surrogate training data for 96% of PropBank rolesets with at least one fully-annotated sentence. This represents 64% of all PropBank rolesets that are used for annotation. Beyond 120 surrogate training examples, F-scores remain around 0.6 before slowly declining after around 700 examples.

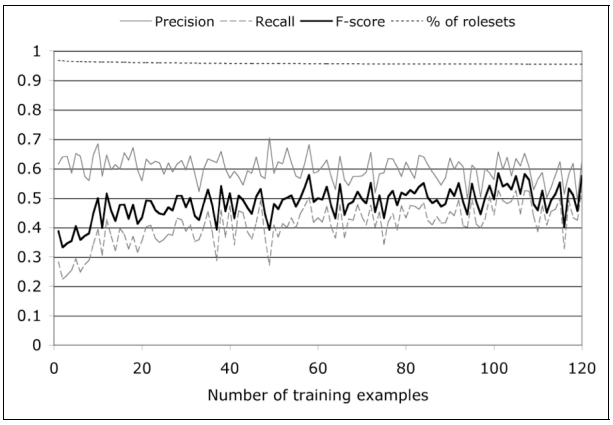


Figure 3. Performance of our semantic role labeling approach on PropBank rolesets using various amounts of surrogate training data

Several interesting comparisons can be made between the results presented in Figure 3 and those in Figure 2, where actual PropBank training data is used instead of surrogate training data. First, the precision obtained with surrogate training data is roughly 10% lower than with real data. Second, the recall performance of surrogate data performs similar to real data at first, but is consistently 10% lower than with real data after the first 50 training examples. Accordingly, F-scores for surrogate training data are 10% lower overall.

Even though the performance obtained using surrogate training data is less than with actual data, there is abundant amounts of it available for most PropBank rolesets. Comparing the "% of rolesets" plots in Figures 2 and 3, the real value of surrogate training data is apparent. Figure 2 suggests that over 20 real training examples are needed to achieve F-scores that are consistently above 0.5, but that less than 20% of PropBank rolesets have this much data available. In contrast, 64% of all PropBank rolesets can achieve this F-score performance with the use of surrogate training data. This percentage increases to 96% if every PropBank roleset is given at least one fully annotated sentence, where all of its numbered arguments are assigned to constituents.

In addition to supplementing the real training data available for existing PropBank rolesets, these results predict the labeling performance that can be obtained by applying this technique to a novel roleset with one fully-annotated training example, e.g. for the verb *yearn*. Using the first 120 surrogate training examples and our simple semantic role labeling approach, we would expect F-scores that are above 0.5, and that using the first 700 would yield F-scores around 0.6.

7 Discussion

The overall performance of our semantic role labeling approach is not competitive with leading contemporary systems, which typically employ support vector machine learning algorithms with syntactic features (Pradhan et al., 2005) or syntactic tree kernels (Moschitti et al., 2006). However, our work highlights a number of characteristics of the semantic role labeling task that will be helpful in improving performance in future systems. Parse tree paths features can be used to achieve high precision in semantic role labeling, but much of this precision may be specific to individual verbs. By generalizing parse tree path features only across syntactically similar verbs, we have shown that the drop in precision can be limited to roughly 10%.

The approach that we describe in this paper is not dependent on the use of PropBank rolesets; any large corpus of semantic role annotations could be

generalized in this manner. In particular, our approach would be applicable to corpora with frame-specific role labels, e.g. FrameNet (Baker et al., 1998). Likewise, our approach to generalizing parse tree path feature across syntactically similar verbs may improve the performance of automated semantic role labeling systems based on FrameNet data. Our work suggests that feature generalization based on verb-similarity may compliment approaches to generalization based on role-similarity (Gildea and Jurafsky, 2002; Baldewein et al., 2004).

There are a number of improvements that could be made to the approach described in this paper. Enhancements to the simple semantic role labeling algorithm would improve the alignment of arguments across rolesets, which would help align rolesets with greater syntactic similarity, as well as improve the performance obtained using the surrogate training data in assigning semantic roles.

This research raises many questions about the relationship between syntactic context and verb semantics. An important area for future research will be to explore the correlation between our distance metric for syntactic similarity and various quantitative measures of semantic similarity (Pedersen, et al., 2004). Particularly interesting would be to explore whether different senses of a given verb exhibited markedly different profiles of syntactic context. A strong syntactic/semantic correlation would suggest that further gains in the use of surrogate annotation data could be gained if syntactic similarity was computed between rolesets rather than their verbs. However, this would first require accurate word-sense disambiguation both for the test sentences as well as for the parsed corpora used to calculate parse tree path frequencies. Alternatively, parse tree path profiles associated with rolesets may be useful for word sense disambiguation, where the probability of a sense is computed as the likelihood that an ambiguous verb's parse tree paths are sampled from the distributions associated with each verb sense. These topics will be the focus of our future work in this area.

Acknowledgments

The project or effort depicted was or is sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM), and that the content or information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

Baker, C., Fillmore, C., and Lowe, J. 1998. The Berkeley FrameNet Project, In Proceedings of COLING-ACL, Montreal.

Baldewein, U., Erk, K., Pado, S., and Prescher, D. 2004. Semantic role labeling with similarity-based generalization using EM-based clustering. Proceedings of Senseval-3, Barcelona.

Charniak, E. 2000. A maximum-entropy-inspired parser, Proceedings NAACL-ANLP, Seattle.

Courtois, B. 2004. Dictionnaires électroniques DELAF anglais et français. In C. Leclère, E. Laporte, M. Piot and M. Silberztein (eds.) Syntax, Lexis and Lexicon-Grammar: Papers in Honour of Maurice Gross. Amsterdam: John Benjamins.

Gildea, D. and Jurafsky, D. 2002. Automatic Labeling of Semantic Roles. Computational Linguistics 28:3, 245-288.

Joanis, E. and Stevenson, S. 2003. A general feature space for automatic verb classification. Proceedings EACL, Budapest.

Levin, B. 1993. English Verb Classes and Alterna-tions: A Preliminary Investigation. Chicago, IL: University of Chicago Press.

Lin, D. 1998. Automatic Retrieval and Clustering of Similar Words. COLING-ACL, Montreal.

Linguistic Data Consortium. 2003. English Gigaword. Catalog number LDC2003T05. Available from LDC at http://www.ldc.upenn.edu.

Moschitti, A., Pighin, D. and Basili, R. 2006. Semantic Role Labeling via Tree Kernel joint inference. Proceedings of CoNLL, New York.

Palmer, M., Gildea, D., and Kingsbury, P. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. Computational Linguistics 31(1):71-106.

Pedersen, T., Patwardhan, S. and Michelizzi, J. 2004. WordNet::Similarity - Measuring the Relatedness of Concepts. Proceedings NAACL-04, Boston, MA.

Pradhan, S., Ward, W., Hacioglu, K., Martin, J., and Jurafsky, D. 2005. Semantic role labeling using different syntactic views. Proceedings ACL-2005, Ann Arbor, MI.

Reynar, J. and Ratnaparkhi, A. 1997. A Maximum Entropy Approach to Identifying Sentence Boundaries. Proceedings of ANLP, Washington, D.C.

A Grammar-driven Convolution Tree Kernel for Semantic Role Classification

Min ZHANG[1] Wanxiang CHE[2] Ai Ti AW[1] Chew Lim TAN[3]
Guodong ZHOU[1,4] Ting LIU[2] Sheng LI[2]

[1]Institute for Infocomm Research
{mzhang, aaiti}@i2r.a-star.edu.sg

[2]Harbin Institute of Technology
{car, tliu}@ir.hit.edu.cn
lisheng@hit.edu.cn

[3]National University of Singapore
tancl@comp.nus.edu.sg

[4] Soochow Univ., China 215006
gdzhou@suda.edu.cn

Abstract

Convolution tree kernel has shown promising results in semantic role classification. However, it only carries out hard matching, which may lead to over-fitting and less accurate similarity measure. To remove the constraint, this paper proposes a grammar-driven convolution tree kernel for semantic role classification by introducing more linguistic knowledge into the standard tree kernel. The proposed grammar-driven tree kernel displays two advantages over the previous one: 1) grammar-driven approximate substructure matching and 2) grammar-driven approximate tree node matching. The two improvements enable the grammar-driven tree kernel explore more linguistically motivated structure features than the previous one. Experiments on the CoNLL-2005 SRL shared task show that the grammar-driven tree kernel significantly outperforms the previous non-grammar-driven one in SRL. Moreover, we present a composite kernel to integrate feature-based and tree kernel-based methods. Experimental results show that the composite kernel outperforms the previously best-reported methods.

1 Introduction

Given a sentence, the task of Semantic Role Labeling (SRL) consists of analyzing the logical forms expressed by some target verbs or nouns and some constituents of the sentence. In particular, for each predicate (target verb or noun) all the constituents in the sentence which fill semantic arguments (roles) of the predicate have to be recognized. Typical semantic roles include *Agent, Patient, Instrument,* etc. and also adjuncts such as *Locative, Temporal, Manner,* and *Cause,* etc. Generally, semantic role identification and classification are regarded as two key steps in semantic role labeling. Semantic role identification involves classifying each syntactic element in a sentence into either a semantic argument or a non-argument while semantic role classification involves classifying each semantic argument identified into a specific semantic role. This paper focuses on semantic role classification task with the assumption that the semantic arguments have been identified correctly.

Both feature-based and kernel-based learning methods have been studied for semantic role classification (Carreras and Màrquez, 2004; Carreras and Màrquez, 2005). In feature-based methods, a flat feature vector is used to represent a predicate-argument structure while, in kernel-based methods, a kernel function is used to measure directly the similarity between two predicate-argument structures. As we know, kernel methods are more effective in capturing structured features. Moschitti (2004) and Che et al. (2006) used a convolution tree kernel (Collins and Duffy, 2001) for semantic role classification. The convolution tree kernel takes sub-tree as its feature and counts the number of common sub-trees as the similarity between two predicate-arguments. This kernel has shown very

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 200–207,
Prague, Czech Republic, June 2007.

promising results in SRL. However, as a general learning algorithm, the tree kernel only carries out hard matching between any two sub-trees without considering any linguistic knowledge in kernel design. This makes the kernel fail to handle similar phrase structures (e.g., "buy a car" *vs*. "buy a *red* car") and near-synonymic grammar tags (e.g., the POS variations between "*high*/JJ degree/NN" [1] and "*higher*/JJR degree/NN") [2]. To some degree, it may lead to over-fitting and compromise performance.

This paper reports our preliminary study in addressing the above issue by introducing more linguistic knowledge into the convolution tree kernel. To our knowledge, this is the first attempt in this research direction. In detail, we propose a grammar-driven convolution tree kernel for semantic role classification that can carry out more linguistically motivated substructure matching. Experimental results show that the proposed method significantly outperforms the standard convolution tree kernel on the data set of the CoNLL-2005 SRL shared task.

The remainder of the paper is organized as follows: Section 2 reviews the previous work and Section 3 discusses our grammar-driven convolution tree kernel. Section 4 shows the experimental results. We conclude our work in Section 5.

2 Previous Work

Feature-based Methods for SRL: most features used in prior SRL research are generally extended from Gildea and Jurafsky (2002), who used a linear interpolation method and extracted basic flat features from a parse tree to identify and classify the constituents in the FrameNet (Baker et al., 1998). Here, the basic features include Phrase Type, Parse Tree Path, and Position. Most of the following work focused on feature engineering (Xue and Palmer, 2004; Jiang et al., 2005) and machine learning models (Nielsen and Pradhan, 2004; Pradhan et al., 2005a). Some other work paid much attention to the robust SRL (Pradhan et al., 2005b) and post inference (Punyakanok et al., 2004). These feature-based methods are considered as the state of the art methods for SRL. However, as we know, the standard flat features are less effective in modeling the syntactic structured information. For example, in SRL, the Parse Tree Path feature is sensitive to small changes of the syntactic structures. Thus, a predicate argument pair will have two different Path features even if their paths differ only for one node. This may result in data sparseness and model generalization problems.

Kernel-based Methods for SRL: as an alternative, kernel methods are more effective in modeling structured objects. This is because a kernel can measure the similarity between two structured objects using the original representation of the objects instead of explicitly enumerating their features. Many kernels have been proposed and applied to the NLP study. In particular, Haussler (1999) proposed the well-known convolution kernels for a discrete structure. In the context of it, more and more kernels for restricted syntaxes or specific domains (Collins and Duffy, 2001; Lodhi et al., 2002; Zelenko et al., 2003; Zhang et al., 2006) are proposed and explored in the NLP domain.

Of special interest here, Moschitti (2004) proposed Predicate Argument Feature (PAF) kernel for SRL under the framework of convolution tree kernel. He selected portions of syntactic parse trees as predicate-argument feature spaces, which include salient substructures of predicate-arguments, to define convolution kernels for the task of semantic role classification. Under the same framework, Che et al. (2006) proposed a hybrid convolution tree kernel, which consists of two individual convolution kernels: a Path kernel and a Constituent Structure kernel. Che et al. (2006) showed that their method outperformed PAF on the CoNLL-2005 SRL dataset.

The above two kernels are special instances of convolution tree kernel for SRL. As discussed in Section 1, convolution tree kernel only carries out hard matching, so it fails to handle similar phrase structures and near-synonymic grammar tags. This paper presents a grammar-driven convolution tree kernel to solve the two problems

3 Grammar-driven Convolution Tree Kernel

3.1 Convolution Tree Kernel

In convolution tree kernel (Collins and Duffy, 2001), a parse tree T is represented by a vector of integer counts of each sub-tree type (regardless of its ancestors): $\phi(T) = (\ldots, \#subtree_i(T), \ldots)$, where

[1] Please refer to http://www.cis.upenn.edu/~treebank/ for the detailed definitions of the grammar tags used in the paper.

[2] Some rewrite rules in English grammar are generalizations of others: for example, "NP→ DET JJ NN" is a specialized version of "NP→ DET NN". The same applies to POS. The standard convolution tree kernel is unable to capture the two cases.

$subtree_i(T)$ is the occurrence number of the i^{th} sub-tree type ($subtree_i$) in T. Since the number of different sub-trees is exponential with the parse tree size, it is computationally infeasible to directly use the feature vector $\phi(T)$. To solve this computational issue, Collins and Duffy (2001) proposed the following parse tree kernel to calculate the dot product between the above high dimensional vectors implicitly.

$$K(T_1,T_2) = <\phi(T_1),\phi(T_2)> = \sum_i \left(\left(\sum_{n_1\in N_1} I_{subtree_i}(n_1)\right)\cdot\left(\sum_{n_2\in N_2} I_{subtree_i}(n_2)\right)\right) = \sum_{n_1\in N_1}\sum_{n_2\in N_2}\Delta(n_1,n_2)$$

where N_1 and N_2 are the sets of nodes in trees T_1 and T_2, respectively, and $I_{subtree_i}(n)$ is a function that is 1 iff the $subtree_i$ occurs with root at node n and zero otherwise, and $\Delta(n_1,n_2)$ is the number of the common *subtrees* rooted at n_1 and n_2, i.e.,

$$\Delta(n_1,n_2) = \sum_i I_{subtree_i}(n_1)\cdot I_{subtree_i}(n_2)$$

$\Delta(n_1,n_2)$ can be further computed efficiently by the following recursive rules:

Rule 1: if the productions (CFG rules) at n_1 and n_2 are different, $\Delta(n_1,n_2)=0$;

Rule 2: else if both n_1 and n_2 are pre-terminals (POS tags), $\Delta(n_1,n_2)=1\times\lambda$;

Rule 3: else,

$$\Delta(n_1,n_2) = \lambda\prod_{j=1}^{nc(n_1)}(1+\Delta(ch(n_1,j),ch(n_2,j))),$$

where $nc(n_1)$ is the child number of n_1, $ch(n,j)$ is the j^{th} child of node n and λ ($0<\lambda<1$) is the decay factor in order to make the kernel value less variable with respect to the *subtree* sizes. In addition, the recursive **Rule 3** holds because given two nodes with the same children, one can construct common sub-trees using these children and common sub-trees of further offspring. The time complexity for computing this kernel is $O(|N_1|\cdot|N_2|)$.

3.2 Grammar-driven Convolution Tree Kernel

This Subsection introduces the two improvements and defines our grammar-driven tree kernel.

Improvement 1: Grammar-driven approximate matching between substructures. The conventional tree kernel requires exact matching between two contiguous phrase structures. This constraint may be too strict. For example, the two phrase structures "NP→DT JJ NN" (NP→*a red car*) and "NP→DT NN" (NP->*a car*) are not identical, thus they contribute nothing to the conventional kernel although they should share the same semantic role given a predicate. In this paper, we propose a grammar-driven approximate matching mechanism to capture the similarity between such kinds of quasi-structures for SRL.

First, we construct reduced rule set by defining optional nodes, for example, "NP->DT [JJ] NP" or "VP-> VB [ADVP] PP", where [*] denotes optional nodes. For convenience, we call "NP-> DT JJ NP" the original rule and "NP->DT [JJ] NP" the reduced rule. Here, we define two grammar-driven criteria to select optional nodes:

1) The reduced rules must be grammatical. It means that the reduced rule should be a valid rule in the original rule set. For example, "NP->DT [JJ] NP" is valid only when "NP->DT NP" is a valid rule in the original rule set while "NP->DT [JJ NP]" may not be valid since "NP->DT" is not a valid rule in the original rule set.

2) A valid reduced rule must keep the head child of its corresponding original rule and has at least two children. This can make the reduced rules retain the underlying semantic meaning of their corresponding original rules.

Given the reduced rule set, we can then formulate the approximate substructure matching mechanism as follows:

$$M(r_1,r_2) = \sum_{i,j}(I_T(T_{r1}^i,T_{r2}^j)\times\lambda_1^{a_i+b_j}) \qquad (1)$$

where r_1 is a production rule, representing a sub-tree of depth one[3], and T_{r1}^i is the i^{th} variation of the sub-tree r_1 by removing one ore more optional nodes[4], and likewise for r_2 and T_{r2}^j. $I_T(\bullet,\bullet)$ is a function that is 1 iff the two sub-trees are identical and zero otherwise. λ_1 ($0\le\lambda_1\le1$) is a small penalty to penal-

[3] Eq.(1) is defined over sub-structure of depth one. The approximate matching between structures of depth more than one can be achieved easily through the matching of sub-structures of depth one in the recursively-defined convolution kernel. We will discuss this issue when defining our kernel.

[4] To make sure that the new kernel is a proper kernel, we have to consider all the possible variations of the original sub-trees. Training program converges only when using a proper kernel.

ize optional nodes and the two parameters a_i and b_j stand for the numbers of occurrence of removed optional nodes in subtrees T_{r1}^i and T_{r2}^j, respectively. $M(r_1, r_2)$ returns the similarity (ie., the kernel value) between the two sub-trees r_1 and r_2 by summing up the similarities between all possible variations of the sub-trees r_1 and r_2.

Under the new approximate matching mechanism, two structures are matchable (but with a small penalty λ_1) if the two structures are identical after removing one or more optional nodes. In this case, the above example phrase structures "NP->*a red car*" and "NP->*a car*" are matchable with a penalty λ_1 in our new kernel. It means that one co-occurrence of the two structures contributes λ_1 to our proposed kernel while it contributes zero to the traditional one. Therefore, by this improvement, our method would be able to explore more linguistically appropriate features than the previous one (which is formulated as $I_T(r_1, r_2)$).

Improvement 2: Grammar-driven tree nodes approximate matching. The conventional tree kernel needs an exact matching between two (terminal/non-terminal) nodes. But, some similar POSs may represent similar roles, such as NN (*dog*) and NNS (*dogs*). In order to capture this phenomenon, we allow approximate matching between node features. The following illustrates some equivalent node feature sets:

- JJ, JJR, JJS
- VB, VBD, VBG, VBN, VBP, VBZ
- ……

where POSs in the same line can match each other with a small penalty $0 \le \lambda_2 \le 1$. We call this case node feature *mutation*. This improvement further generalizes the conventional tree kernel to get better coverage. The approximate node matching can be formulated as:

$$M(f_1, f_2) = \sum_{i,j} (I_f(f_1^i, f_2^j) \times \lambda_2^{a_i + b_j}) \qquad (2)$$

where f_1 is a node feature, f_1^i is the i^{th} mutation of f_1 and a_i is 0 iff f_1^i and f_1 are identical and 1 otherwise, and likewise for f_2. $I_f(\bullet, \bullet)$ is a function that is 1 iff the two features are identical and zero otherwise. Eq. (2) sums over all combinations of feature mutations as the node feature similarity. The same as Eq. (1), the reason for taking all the possibilities into account in Eq. (2) is to make sure that the new kernel is a proper kernel.

The above two improvements are grammar-driven, i.e., the two improvements retain the underlying linguistic grammar constraints and keep semantic meanings of original rules.

The Grammar-driven Kernel Definition: Given the two improvements discussed above, we can define the new kernel by beginning with the feature vector representation of a parse tree *T* as follows:

$$\phi'(T) = (\#\, subtree_1(T), \ldots, \#\, subtree_n(T))$$

where $\#\, subtree_i(T)$ is the occurrence number of the i^{th} sub-tree type ($subtree_i$) in *T*. Please note that, different from the previous tree kernel, here we loosen the condition for the occurrence of a subtree by allowing both original and reduced rules (**Improvement 1**) and node feature mutations (**Improvement 2**). In other words, we modify the criteria by which a subtree is said to occur. For example, one occurrence of the rule "NP->DT JJ NP" shall contribute 1 times to the feature "NP->DT JJ NP" and λ_1 times to the feature "NP->DT NP" in the new kernel while it only contributes 1 times to the feature "NP->DT JJ NP" in the previous one. Now we can define the new grammar-driven kernel $K_G(T_1, T_2)$ as follows:

$$\begin{aligned} K_G(T_1, T_2) &= \langle \phi'(T_1), \phi'(T_2) \rangle \\ &= \sum_i \left(\left(\sum_{n_1 \in N_1} I'_{subtree_i}(n_1) \right) \cdot \left(\sum_{n_2 \in N_2} I'_{subtree_i}(n_2) \right) \right) \\ &= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \Delta'(n_1, n_2) \end{aligned} \qquad (3)$$

where N_1 and N_2 are the sets of nodes in trees T_1 and T_2, respectively. $I'_{subtree_i}(n)$ is a function that is $\lambda_1^a \cdot \lambda_2^b$ iff the $subtree_i$ occurs with root at node *n* and zero otherwise, where a and b are the numbers of removed optional nodes and mutated node features, respectively. $\Delta'(n_1, n_2)$ is the number of the common *subtrees* rooted at n_1 and n_2, i.e. ,

$$\Delta'(n_1, n_2) = \sum_i I'_{subtree_i}(n_1) \cdot I'_{subtree_i}(n_2) \qquad (4)$$

Please note that the value of $\Delta'(n_1, n_2)$ is no longer an integer as that in the conventional one since optional nodes and node feature mutations are considered in the new kernel. $\Delta'(n_1, n_2)$ can be further computed by the following recursive rules:

Rule A: if n_1 and n_2 are pre-terminals, then:

$$\Delta'(n_1, n_2) = \lambda \times M(f_1, f_2) \quad (5)$$

where f_1 and f_2 are features of nodes n_1 and n_2 respectively, and $M(f_1, f_2)$ is defined at Eq. (2).

Rule B: else if both n_1 and n_2 are the same non-terminals, then generate all variations of the subtrees of *depth one* rooted by n_1 and n_2 (denoted by T_{n1} and T_{n2} respectively) by removing different optional nodes, then:

$$\Delta'(n_1, n_2) = \lambda \times \sum_{i,j} (I_T(T_{n1}^i, T_{n2}^j) \times \lambda_1^{a_i + b_j} \times \prod_{k=1}^{nc(n_1,i)} (1 + \Delta'(ch(n_1, i, k), ch(n_2, j, k))) \quad (6)$$

where

- T_{n1}^i and T_{n2}^j stand for the i^{th} and j^{th} variations in sub-tree set T_{n1} and T_{n2}, respectively.
- $I_T(\bullet, \bullet)$ is a function that is 1 iff the two sub-trees are identical and zero otherwise.
- a_i and b_j stand for the number of removed optional nodes in subtrees T_{n1}^i and T_{n2}^j, respectively.
- $nc(n_1, i)$ returns the child number of n_1 in its i^{th} subtree variation T_{n1}^i.
- $ch(n_1, i, k)$ is the k^{th} child of node n_1 in its i^{th} variation subtree T_{n1}^i, and likewise for $ch(n_2, j, k)$.
- Finally, the same as the previous tree kernel, λ ($0 < \lambda < 1$) is the decay factor (see the discussion in Subsection 3.1).

Rule C: else $\Delta'(n_1, n_2) = 0$

Rule A accounts for **Improvement** 2 while **Rule B** accounts for **Improvement** 1. In **Rule B**, Eq. (6) is able to carry out multi-layer sub-tree approximate matching due to the introduction of the recursive part while Eq. (1) is only effective for sub-trees of depth one. Moreover, we note that Eq. (4) is a convolution kernel according to the definition and the proof given in Haussler (1999), and Eqs (5) and (6) reformulate Eq. (4) so that it can be computed efficiently, in this way, our kernel defined by Eq (3) is also a valid convolution kernel. Finally, let us study the computational issue of the new convolution tree kernel. Clearly, computing Eq. (6) requires exponential time in its worst case. However, in practice, it may only need $O(|N_1| \cdot |N_2|)$. This is because there are only 9.9% rules (647 out of the total 6,534 rules in the parse trees) have optional nodes and most of them have only one optional node. In fact, the actual running time is even much less and is close to linear in the size of the trees since $\Delta'(n_1, n_2) = 0$ holds for many node pairs (Collins and Duffy, 2001). In theory, we can also design an efficient algorithm to compute Eq. (6) using a dynamic programming algorithm (Moschitti, 2006). We just leave it for our future work.

3.3 Comparison with previous work

In above discussion, we show that the conventional convolution tree kernel is a special case of the grammar-driven tree kernel. From kernel function viewpoint, our kernel can carry out not only exact matching (as previous one described by **Rules 2** and **3** in Subsection 3.1) but also approximate matching (Eqs. (5) and (6) in Subsection 3.2). From feature exploration viewpoint, although they explore the same sub-structure feature space (defined recursively by the phrase parse rules), their feature values are different since our kernel captures the structure features in a more linguistically appropriate way by considering more linguistic knowledge in our kernel design.

Moschitti (2006) proposes a partial tree (PT) kernel which can carry out partial matching between sub-trees. The PT kernel generates a much larger feature space than both the conventional and the grammar-driven kernels. In this point, one can say that the grammar-driven tree kernel is a specialization of the PT kernel. However, the important difference between them is that the PT kernel is not grammar-driven, thus many non-linguistically motivated structures are matched in the PT kernel. This may potentially compromise the performance since some of the over-generated features may possibly be noisy due to the lack of linguistic interpretation and constraint.

Kashima and Koyanagi (2003) proposed a convolution kernel over labeled order trees by generalizing the standard convolution tree kernel. The labeled order tree kernel is much more flexible than the PT kernel and can explore much larger sub-tree features than the PT kernel. However, the same as the PT kernel, the labeled order tree kernel is not grammar-driven. Thus, it may face the same issues

(such as over-generated features) as the PT kernel when used in NLP applications.

Shen el al. (2003) proposed a lexicalized tree kernel to utilize LTAG-based features in parse reranking. Their methods need to obtain a LTAG derivation tree for each parse tree before kernel calculation. In contrast, we use the notion of optional arguments to define our grammar-driven tree kernel and use the empirical set of CFG rules to determine which arguments are optional.

4 Experiments

4.1 Experimental Setting

Data: We use the CoNLL-2005 SRL shared task data (Carreras and Màrquez, 2005) as our experimental corpus. The data consists of sections of the Wall Street Journal part of the Penn TreeBank (Marcus et al., 1993), with information on predicate-argument structures extracted from the PropBank corpus (Palmer et al., 2005). As defined by the shared task, we use sections 02-21 for training, section 24 for development and section 23 for test. There are 35 roles in the data including 7 Core (A0–A5, AA), 14 Adjunct (AM-) and 14 Reference (R-) arguments. Table 1 lists counts of sentences and arguments in the three data sets.

	Training	Development	Test
Sentences	39,832	1,346	2,416
Arguments	239,858	8,346	14,077

Table 1: Counts on the data set

We assume that the semantic role identification has been done correctly. In this way, we can focus on the classification task and evaluate it more accurately. We evaluate the performance with Accuracy. SVM (Vapnik, 1998) is selected as our classifier and the *one vs. others* strategy is adopted and the one with the largest margin is selected as the final answer. In our implementation, we use the binary SVMLight (Joachims, 1998) and modify the Tree Kernel Tools (Moschitti, 2004) to a grammar-driven one.

Kernel Setup: We use the Constituent, Predicate, and Predicate-Constituent related features, which are reported to get the best-reported performance (Pradhan et al., 2005a), as the baseline features. We use Che et al. (2006)'s hybrid convolution tree kernel (the best-reported method for kernel-based SRL) as our baseline kernel. It is defined as $K_{hybrid} = \theta K_{path} + (1-\theta)K_{cs}$ $(0 \le \theta \le 1)$ (for the detailed definitions of K_{path} and K_{cs}, please refer to Che et al. (2006)). Here, we use our grammar-driven tree kernel to compute K_{path} and K_{cs}, and we call it grammar-driven hybrid tree kernel while Che et al. (2006)'s is non-grammar-driven hybrid convolution tree kernel.

We use a greedy strategy to fine-tune parameters. Evaluation on the development set shows that our kernel yields the best performance when λ (decay factor of tree kernel), λ_1 and λ_2 (two penalty factors for the grammar-driven kernel), θ (hybrid kernel parameter) and c (a SVM training parameter to balance training error and margin) are set to 0.4, 0.6, 0.3, 0.6 and 2.4, respectively. For other parameters, we use default setting. In the CoNLL 2005 benchmark data, we get 647 rules with optional nodes out of the total 6,534 grammar rules and define three equivalent node feature sets as below:

- JJ, JJR, JJS
- RB, RBR, RBS
- NN, NNS, NNP, NNPS, NAC, NX

Here, the verb feature set "VB, VBD, VBG, VBN, VBP, VBZ" is removed since the voice information is very indicative to the arguments of ARG0 (Agent, operator) and ARG1 (Thing operated).

Methods	Accuracy (%)
Baseline: Non-grammar-driven	85.21
+Approximate Node Matching	86.27
+Approximate Substructure Matching	87.12
Ours: Grammar-driven Substructure and Node Matching	87.96
Feature-based method with polynomial kernel (d = 2)	89.92

Table 2: Performance comparison

4.2 Experimental Results

Table 2 compares the performances of different methods on the test set. First, we can see that the new grammar-driven hybrid convolution tree kernel significantly outperforms (χ^2 test with p=0.05) the

non-grammar one with an absolute improvement of 2.75 (87.96-85.21) percentage, representing a relative error rate reduction of 18.6% (2.75/(100-85.21)) . It suggests that 1) the linguistically motivated structure features are very useful for semantic role classification and 2) the grammar-driven kernel is much more effective in capturing such kinds of features due to the consideration of linguistic knowledge. Moreover, Table 2 shows that 1) both the grammar-driven approximate node matching and the grammar-driven approximate substructure matching are very useful in modeling syntactic tree structures for SRL since they contribute relative error rate reduction of 7.2% ((86.27-85.21)/(100-85.21)) and 12.9% ((87.12-85.21)/(100-85.21)), respectively; 2) the grammar-driven approximate substructure matching is more effective than the grammar-driven approximate node matching. However, we find that the performance of the grammar-driven kernel is still a bit lower than the feature-based method. This is not surprising since tree kernel methods only focus on modeling tree structure information. In this paper, it captures the syntactic parse tree structure features only while the features used in the feature-based methods cover more knowledge sources.

In order to make full use of the syntactic structure information and the other useful diverse flat features, we present a composite kernel to combine the grammar-driven hybrid kernel and feature-based method with polynomial kernel:

$$K_{comp} = \gamma K_{hybrid} + (1-\gamma) K_{poly} \qquad (0 \le \gamma \le 1)$$

Evaluation on the development set shows that the composite kernel yields the best performance when γ is set to 0.3. Using the same setting, the system achieves the performance of 91.02% in Accuracy in the same test set. It shows statistically significant improvement (χ^2 test with p= 0.10) over using the standard features with the polynomial kernel ($\gamma = 0$, Accuracy = 89.92%) and using the grammar-driven hybrid convolution tree kernel ($\gamma = 1$, Accuracy = 87.96%). The main reason is that the tree kernel can capture effectively more structure features while the standard flat features can cover some other useful features, such as Voice, SubCat, which are hard to be covered by the tree kernel. The experimental results suggest that these two kinds of methods are complementary to each other.

In order to further compare with other methods, we also do experiments on the dataset of English PropBank I (LDC2004T14). The training, development and test sets follow the conventional split of Sections 02-21, 00 and 23. Table 3 compares our method with other previously best-reported methods with the same setting as discussed previously. It shows that our method outperforms the previous best-reported one with a relative error rate reduction of 10.8% (0.97/(100-91)). This further verifies the effectiveness of the grammar-driven kernel method for semantic role classification.

Method	Accuracy (%)
Ours (Composite Kernel)	**91.97**
Moschitti (2006): PAF kernel only	87.7
Jiang et al. (2005): feature based	90.50
Pradhan et al. (2005a): feature based	91.0

Table 3: Performance comparison between our method and previous work

Method	Training Time	
	4 Sections	19 Sections
Ours: grammar-driven tree kernel	~8.1 hours	~7.9 days
Moschitti (2006): non-grammar-driven tree kernel	~7.9 hours	~7.1 days

Table 4: Training time comparison

Table 4 reports the training times of the two kernels. We can see that 1) the two kinds of convolution tree kernels have similar computing time. Although computing the grammar-driven one requires exponential time in its worst case, however, in practice, it may only need $O(|N_1| \cdot |N_2|)$ or linear and 2) it is very time-consuming to train a SVM classifier in a large dataset.

5 Conclusion and Future Work

In this paper, we propose a novel grammar-driven convolution tree kernel for semantic role classification. More linguistic knowledge is considered in the new kernel design. The experimental results verify that the grammar-driven kernel is more effective in capturing syntactic structure features than the previous convolution tree kernel because it allows grammar-driven approximate matching of substructures and node features. We also discuss the criteria to determine the optional nodes in a

CFG rule in defining our grammar-driven convolution tree kernel.

The extension of our work is to improve the performance of the entire semantic role labeling system using the grammar-driven tree kernel, including all four stages: pruning, semantic role identification, classification and post inference. In addition, a more interesting research topic is to study how to integrate linguistic knowledge and tree kernel methods to do feature selection for tree kernel-based NLP applications (Suzuki et al., 2004). In detail, a linguistics and statistics-based theory that can suggest the effectiveness of different substructure features and whether they should be generated or not by the tree kernels would be worked out.

References

C. F. Baker, C. J. Fillmore, and J. B. Lowe. 1998. *The Berkeley FrameNet Project*. COLING-ACL-1998

Xavier Carreras and Lluıs Màrquez. 2004. Introduction *to the CoNLL-2004 shared task: Semantic role labeling*. CoNLL-2004

Xavier Carreras and Lluıs Màrquez. 2005. Introduction *to the CoNLL-2005 shared task: Semantic role labeling*. CoNLL-2005

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

Wanxiang Che, Min Zhang, Ting Liu and Sheng Li. 2006. *A hybrid convolution tree kernel for semantic role labeling*. COLING-ACL-2006(poster)

Michael Collins and Nigel Duffy. 2001. *Convolution kernels for natural language*. NIPS-2001

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

David Haussler. 1999. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10

Zheng Ping Jiang, Jia Li and Hwee Tou Ng. 2005. *Semantic argument classification exploiting argument interdependence*. IJCAI-2005

T. Joachims. 1998. *Text Categorization with Support Vecor Machine: learning with many relevant features*. ECML-1998

Kashima H. and Koyanagi T. 2003. *Kernels for Semi-Structured Data*. ICML-2003

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini and Chris Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research,* 2:419–444

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

Alessandro Moschitti. 2004. *A study on convolution kernels for shallow statistic parsing*. ACL-2004

Alessandro Moschitti. 2006. *Syntactic kernels for natural language learning: the semantic role labeling case*. HLT-NAACL-2006 (short paper)

Rodney D. Nielsen and Sameer Pradhan. 2004. *Mixing weak learners in semantic parsing*. EMNLP-2004

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

Sameer Pradhan, Kadri Hacioglu, Valeri Krugler, Wayne Ward, James H. Martin and Daniel Jurafsky. 2005a. Support vector learning for semantic argument classification. *Journal of Machine Learning*

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin and Daniel Jurafsky. 2005b. *Semantic role labeling using different syntactic views.* ACL-2005

Vasin Punyakanok, Dan Roth, Wen-tau Yih and Dav Zimak. 2004. *Semantic role labeling via integer linear programming inference*. COLING-2004

Vasin Punyakanok, Dan Roth and Wen Tau Yih. 2005. *The necessity of syntactic parsing for semantic role labeling*. IJCAI-2005

Libin Shen, Anoop Sarkar and A. K. Joshi. 2003. *Using LTAG based features in parse reranking*. EMNLP-03

Jun Suzuki, Hideki Isozaki and Eisaku Maede. 2004. *Convolution kernels with feature selection for Natural Language processing tasks*. ACL-2004

Vladimir N. Vapnik. 1998. *Statistical Learning Theory.* Wiley

Nianwen Xue and Martha Palmer. 2004. *Calibrating features for semantic role labeling*. EMNLP-2004

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Machine Learning Research,* 3:1083–1106

Min Zhang, Jie Zhang, Jian Su and Guodong Zhou. 2006. *A Composite Kernel to Extract Relations between Entities with both Flat and Structured Features*. COLING-ACL-2006

Learning Predictive Structures for Semantic Role Labeling of NomBank

Chang Liu and **Hwee Tou Ng**
Department of Computer Science
National University of Singapore
3 Science Drive 2, Singapore 117543
{liuchan1, nght}@comp.nus.edu.sg

Abstract

This paper presents a novel application of Alternating Structure Optimization (ASO) to the task of Semantic Role Labeling (SRL) of noun predicates in NomBank. ASO is a recently proposed linear multi-task learning algorithm, which extracts the common structures of multiple tasks to improve accuracy, via the use of auxiliary problems. In this paper, we explore a number of different auxiliary problems, and we are able to significantly improve the accuracy of the NomBank SRL task using this approach. To our knowledge, our proposed approach achieves the highest accuracy published to date on the English NomBank SRL task.

1 Introduction

The task of Semantic Role Labeling (SRL) is to identify predicate-argument relationships in natural language texts in a domain-independent fashion. In recent years, the availability of large human-labeled corpora such as PropBank (Palmer et al., 2005) and FrameNet (Baker et al., 1998) has made possible a statistical approach of identifying and classifying the arguments of verbs in natural language texts. A large number of SRL systems have been evaluated and compared on the standard data set in the CoNLL shared tasks (Carreras and Marquez, 2004; Carreras and Marquez, 2005), and many systems have performed reasonably well. Compared to the previous CoNLL shared tasks (noun phrase bracketing, chunking, clause identification, and named entity recognition), SRL represents a significant step towards processing the semantic content of natural language texts.

Although verbs are probably the most obvious predicates in a sentence, many nouns are also capable of having complex argument structures, often with much more flexibility than its verb counterpart. For example, compare *affect* and *effect*:

> [$_{subj}$ Auto prices] [$_{arg-ext}$ greatly] [$_{pred}$ affect] [$_{obj}$ the PPI].
>
> [$_{subj}$ Auto prices] have a [$_{arg-ext}$ big] [$_{pred}$ effect] [$_{obj}$ on the PPI].
>
> The [$_{pred}$ effect] [$_{subj}$ of auto prices] [$_{obj}$ on the PPI] is [$_{arg-ext}$ big].
>
> [$_{subj}$ The auto prices'] [$_{pred}$ effect] [$_{obj}$ on the PPI] is [$_{arg-ext}$ big].

The arguments of noun predicates can often be more easily omitted compared to the verb predicates:

> The [$_{pred}$ effect] [$_{subj}$ of auto prices] is [$_{arg-ext}$ big].
>
> The [$_{pred}$ effect] [$_{obj}$ on the PPI] is [$_{arg-ext}$ big].
>
> The [$_{pred}$ effect] is [$_{arg-ext}$ big].

With the recent release of NomBank (Meyers et al., 2004), it becomes possible to apply machine learning techniques to the task. So far we are aware of only one English NomBank-based SRL system (Jiang and Ng, 2006), which uses the maximum entropy classifier, although similar efforts are reported on the Chinese NomBank by (Xue, 2006)

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 208–215,
Prague, Czech Republic, June 2007.

and on FrameNet by (Pradhan et al., 2004) using a small set of hand-selected nominalizations. Noun predicates also appear in FrameNet semantic role labeling (Gildea and Jurafsky, 2002), and many FrameNet SRL systems are evaluated in Senseval-3 (Litkowski, 2004).

Semantic role labeling of NomBank is a multi-class classification problem by nature. Using the *one-vs-all* arrangement, that is, one binary classifier for each possible outcome, the SRL task can be treated as multiple binary classification problems. In the latter view, we are presented with the opportunity to exploit the *common structures* of these related problems. This is known as *multi-task learning* in the machine learning literature (Caruana, 1997; Ben-David and Schuller, 2003; Evgeniou and Pontil, 2004; Micchelli and Pontil, 2005; Maurer, 2006).

In this paper, we apply *Alternating Structure Optimization* (ASO) (Ando and Zhang, 2005a) to the semantic role labeling task on NomBank. ASO is a recently proposed linear multi-task learning algorithm based on empirical risk minimization. The method requires the use of multiple auxiliary problems, and its effectiveness may vary depending on the specific auxiliary problems used. ASO has been shown to be effective on the following natural language processing tasks: text categorization, named entity recognition, part-of-speech tagging, and word sense disambiguation (Ando and Zhang, 2005a; Ando and Zhang, 2005b; Ando, 2006).

This paper makes two significant contributions. First, we present a novel application of ASO to the SRL task on NomBank. We explore the effect of different auxiliary problems, and show that learning predictive structures with ASO results in significantly improved SRL accuracy. Second, we achieve accuracy higher than that reported in (Jiang and Ng, 2006) and advance the state of the art in SRL research.

The rest of this paper is organized as follows. We give an overview of NomBank and ASO in Sections 2 and 3 respectively. The baseline linear classifier is described in detail in Section 4, followed by the description of the ASO classifier in Section 5, where we focus on exploring different auxiliary problems. We provide discussions in Section 6, present related work in Section 7, and conclude in Section 8.

2 NomBank

NomBank annotates the set of arguments of noun predicates, just as PropBank annotates the arguments of verb predicates. As many noun predicates are nominalizations (e.g., *replacement* vs. *replace*), the same *frames* are shared with PropBank as much as possible, thus achieving some consistency with the latter regarding the accepted arguments and the meanings of each label.

Unlike in PropBank, arguments in NomBank can overlap with each other and with the predicate. For example:

> [$_{location}$ U.S.] [$_{pred,subj,obj}$ steelmakers] have supplied the steel.

Here the predicate *make* has subject *steelmakers* and object *steel*, analogous to *Steelmakers make steel.* The difference is that here *make* and *steel* are both part of the word *steelmaker*.

Each argument in NomBank is given one or more labels, out of the following 20: ARG0, ARG1, ARG2, ARG3, ARG4, ARG5, ARG8, ARG9, ARGM-ADV, ARGM-CAU, ARGM-DIR, ARGM-DIS, ARGM-EXT, ARGM-LOC, ARGM-MNR, ARGM-MOD, ARGM-NEG, ARGM-PNC, ARGM-PRD, and ARGM-TMP. Thus, the above sentence is annotated in NomBank as:

> [$_{\text{ARGM-LOC}}$ U.S.] [$_{\text{PRED,ARG0,ARG1}}$ steelmakers] have supplied the steel.

3 Alternating structure optimization

This section gives a brief overview of ASO as implemented in this work. For a more complete description, see (Ando and Zhang, 2005a).

3.1 Multi-task linear classifier

Given a set of training samples consisting of n feature vectors and their corresponding binary labels, $\{\mathbf{X}_i, Y_i\}$ for $i \in \{1, \ldots, n\}$ where each $\mathbf{X}_i$ is a p-dimensional vector, a binary linear classifier attempts to approximate the unknown relation by $Y_i = \mathbf{u}^{\mathrm{T}}\mathbf{X}_i$. The outcome is considered +1 if $\mathbf{u}^{\mathrm{T}}\mathbf{X}$ is positive, or –1 otherwise. A well-established way to find the *weight vector* $\mathbf{u}$ is *empirical risk minimization* with *least square regularization*:

$$\hat{\mathbf{u}} = \arg\min_{\mathbf{u}} \frac{1}{n} \sum_{i=1}^{n} L\big(\mathbf{u}^{\mathrm{T}}\mathbf{X}_i, Y_i\big) + \lambda \|\mathbf{u}\|^2 \quad (1)$$

Function $L(p, y)$ is known as the *loss function*. It encodes the penalty for a given discrepancy between the predicted label and the true label. In this work, we use a modification of Huber's robust loss function, similar to that used in (Ando and Zhang, 2005a):

$$L(p,y) = \begin{cases} -4py & \text{if } py < -1 \\ (1-py)^2 & \text{if } -1 \le py < 1 \\ 0 & \text{if } py \ge 1 \end{cases} \quad (2)$$

We fix the regularization parameter λ to 10^{-4}, similar to that used in (Ando and Zhang, 2005a). The expression $\|\mathbf{u}\|^2$ is defined as $\sum_{i=1}^{p} \mathbf{u}_p^2$.

When m binary classification problems are to be solved together, a $h \times p$ matrix Θ may be used to capture the *common structures* of the m weight vectors $\mathbf{u}_l$ for $l \in \{1, \ldots, m\}$ $(h \le m)$. We mandate that the rows of Θ be orthonormal, i.e., $\Theta\Theta^{\mathrm{T}} = I_{h \times h}$. The h rows of Θ represent the h most significant components shared by all the $\mathbf{u}$'s. This relationship is modeled by

$$\mathbf{u}_l = \mathbf{w}_l + \Theta^{\mathrm{T}} \mathbf{v}_l \quad (3)$$

The parameters $[\{\mathbf{w}_l, \mathbf{v}_l\}, \Theta]$ may then be found by *joint empirical risk minimization* over all the m problems, i.e., their values should minimize the combined empirical risk:

$$\sum_{l=1}^{m} \left(\frac{1}{n} \sum_{i=1}^{n} L\Big((\mathbf{w}_l + \Theta^{\mathrm{T}} \mathbf{v}_l)^{\mathrm{T}} \mathbf{X}_i^l, Y_i^l \Big) + \lambda \|\mathbf{w}_l\|^2 \right) \quad (4)$$

3.2 The ASO algorithm

An important observation in (Ando and Zhang, 2005a) is that the binary classification problems used to derive Θ are not necessarily those problems we are aiming to solve. In fact, new problems can be invented for the sole purpose of obtaining a better Θ. Thus, we distinguish between two types of problems in ASO: *auxiliary problems*, which are used to obtain Θ, and *target problems*, which are the problems we are aiming to solve[1].

For instance, in the argument identification task, the only target problem is to identify arguments vs. non-arguments, whereas in the argument classification task, there are 20 binary target problems, one to identify each of the 20 labels (ARG0, ARG1, ...).

The target problems can also be used as an auxiliary problem. In addition, we can *invent* new auxiliary problems, e.g., in the argument identification stage, we can predict whether there are three words between the constituent and the predicate using the features of argument identification.

Assuming there are k target problems and m auxiliary problems, it is shown in (Ando and Zhang, 2005a) that by performing one round of minimization, an approximate solution of Θ can be obtained from (4) by the following algorithm:

1. For each of the m *auxiliary problems*, learn $\mathbf{u}_l$ as described by (1).

2. Find $U = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m]$, a $p \times m$ matrix. This is a simplified version of the definition in (Ando and Zhang, 2005a), made possible because the same λ is used for all auxiliary problems.

3. Perform Singular Value Decomposition (SVD) on U: $U = V_1 D V_2^{\mathrm{T}}$, where V_1 is a $p \times m$ matrix. The first h columns of V_1 are stored as rows of Θ.

4. Given Θ, we learn $\mathbf{w}$ and $\mathbf{v}$ for each of the k *target problems* by minimizing the empirical risk of the associated training samples:

$$\frac{1}{n} \sum_{i=1}^{n} L\Big((\mathbf{w} + \Theta^{\mathrm{T}} \mathbf{v})^{\mathrm{T}} \mathbf{X}_i, Y_i \Big) + \lambda \|\mathbf{w}\|^2 \quad (5)$$

5. The weight vector of each target problem can be found by:

$$\mathbf{u} = \mathbf{w} + \Theta^{\mathrm{T}} \mathbf{v} \quad (6)$$

By choosing a convex loss function, e.g., (2), steps 1 and 4 above can be formulated as convex optimization problems and are efficiently solvable.

The procedure above can be considered as a Principal Component Analysis in the predictor space. Step (3) above extracts the most significant components shared by the predictors of the auxiliary problems and hopefully, by the predictors of the target

[1] Note that this definition deviates slightly from the one in (Ando and Zhang, 2005a). We find the definition here more convenient for our subsequent discussion.

problems as well. The hint of potential significant components helps (5) to outperform the simple linear predictor (1).

4 Baseline classifier

The SRL task is typically separated into two stages: argument identification and argument classification. During the identification stage, each constituent in a sentence's parse tree is labeled as either argument or non-argument. During the classification stage, each argument is given one of the 20 possible labels (ARG0, ARG1, ...). The linear classifier described by (1) is used as the baseline in both stages. For comparison, the F1 scores of a maximum entropy classifier are also reported here.

4.1 Argument identification

Eighteen *baseline features* and six *additional features* are proposed in (Jiang and Ng, 2006) for NomBank argument identification. As the improvement of the F1 score due to the additional features is not statistically significant, we use the set of eighteen baseline features for simplicity. These features are reproduced in Table 1 for easy reference.

Unlike in (Jiang and Ng, 2006), we do not prune arguments dominated by other arguments or those that overlap with the predicate in the training data. Accordingly, we do not maximize the probability of the entire labeled parse tree as in (Toutanova et al., 2005). After the features of every constituent are extracted, each constituent is simply classified independently as either argument or non-argument.

The linear classifier described above is trained on sections 2 to 21 and tested on section 23. A maximum entropy classifier is trained and tested in the same manner. The F1 scores are presented in the first row of Table 3, in columns *linear* and *maxent* respectively. The *J&N* column presents the result reported in (Jiang and Ng, 2006) using both baseline and additional features. The last column *aso* presents the best result from this work, to be explained in Section 5.

4.2 Argument classification

In NomBank, some constituents have more than one label. For simplicity, we always assign exactly one label to each identified argument in this step. For the 0.16% arguments with multiple labels in the training data, we pick the first and discard the rest. (Note that the same is *not* done on the test data.)

1	pred	the stemmed predicate
2	subcat	grammar rule that expands the predicate P's parent
3	ptype	syntactic category (phrase type) of the constituent C
4	hw	syntactic head word of C
5	path	syntactic path from C to P
6	position	whether C is to the left/right of or overlaps with P
7	firstword	first word spanned by C
8	lastword	last word spanned by C
9	lsis.ptype	phrase type of left sister
10	rsis.hw	right sister's head word
11	rsis.hw.pos	POS of right sister's head word
12	parent.ptype	phrase type of parent
13	parent.hw	parent's head word
14	partialpath	path from C to the lowest common ancestor with P
15	ptype & length of path	
16	pred & hw	
17	pred & path	
18	pred & position	

Table 1: Features used in argument identification

A diverse set of 28 features is used in (Jiang and Ng, 2006) for argument classification. In this work, the number of features is pruned to 11, so that we can work with reasonably many auxiliary problems in later experiments with ASO.

To find a smaller set of effective features, we start with all the features considered in (Jiang and Ng, 2006), in (Xue and Palmer, 2004), and various combinations of them, for a total of 52 features. These features are then pruned by the following algorithm:

1. For each feature in the current feature set, do step (2).
2. Remove the selected feature from the feature set. Obtain the F1 score of the remaining features when applied to the argument classification task, on development data section 24 with gold identification.
3. Select the highest of all the scores obtained in

1	position	to the left/right of or overlaps with the predicate
2	ptype	syntactic category (phrase type) of the constituent C
3	firstword	first word spanned by C
4	lastword	last word spanned by C
5	rsis.ptype	phrase type of right sister
6	nomtype	NOM-TYPE of predicate supplied by NOMLEX dictionary
7	predicate & ptype	
8	predicate & lastword	
9	morphed predicate stem & head word	
10	morphed predicate stem & position	
11	nomtype & position	

Table 2: Features used in argument classification

step (2). The corresponding feature is removed from the current feature set if its F1 score is the same as or higher than the F1 score of retaining all features.

4. Repeat steps (1)-(3) until the F1 score starts to drop.

The 11 features so obtained are presented in Table 2. Using these features, a linear classifier and a maximum entropy classifier are trained on sections 2 to 21, and tested on section 23. The F1 scores are presented in the second row of Table 3, in columns *linear* and *maxent* respectively. The *J&N* column presents the result reported in (Jiang and Ng, 2006).

4.3 Further experiments and discussion

In the combined task, we run the identification task with gold parse trees, and then the classification task with the output of the identification task. This way the combined effect of errors from both stages on the final classification output can be assessed. The scores of this complete SRL system are presented in the third row of Table 3.

To test the performance of the combined task on automatic parse trees, we employ two different configurations. First, we train the various classifiers on sections 2 to 21 using gold argument labels and *automatic parse trees* produced by Charniak's reranking parser (Charniak and Johnson, 2005), and test them on section 23 with automatic parse trees. This is the same configuration as reported in (Pradhan et al., 2005; Jiang and Ng, 2006). The scores are presented in the fourth row *auto parse (t&t)* in Table 3.

Next, we train the various classifiers on sections 2 to 21 using gold argument labels and *gold parse trees*. To minimize the discrepancy between gold and automatic parse trees, we remove all the nodes in the gold trees whose POS are *-NONE-*, as they do not span any word and are thus never generated by the automatic parser. The resulting classifiers are then tested on section 23 using automatic parse trees. The scores are presented in the last row *auto parse (test)* of Table 3. We note that *auto parse (test)* consistently outperforms *auto parse (t&t)*.

We believe that *auto parse (test)* is a more realistic setting in which to test the performance of SRL on automatic parse trees. When presented with some previously unseen test data, we are forced to rely on its automatic parse trees. However, for the best results we should take advantage of gold parse trees whenever possible, including those of the *labeled* training data.

	J&N	maxent	linear	aso
identification	82.50	83.58	81.34	85.32
classification	87.80	88.35	87.86	89.17
combined	72.73	75.35	72.63	77.04
auto parse (t&t)	69.14	69.61	67.38	72.11
auto parse (test)	-	71.19	69.05	72.83

Table 3: F1 scores of various classifiers on NomBank SRL

Our maximum entropy classifier consistently outperforms (Jiang and Ng, 2006), which also uses a maximum entropy classifier. The primary difference is that we use a later version of NomBank (September 2006 release vs. September 2005 release). In addition, we use somewhat different features and treat overlapping arguments differently.

5 Applying ASO to SRL

Our ASO classifier uses the same features as the baseline linear classifier. The defining characteristic, and also the major challenge in successfully applying the ASO algorithm is to find related auxiliary problems that can reveal common structures shared

with the target problem. To organize our search for good auxiliary problems for SRL, we separate them into two categories, *unobservable auxiliary problems* and *observable auxiliary problems*.

5.1 Unobservable auxiliary problems

Unobservable auxiliary problems are problems whose true outcome cannot be observed from a raw text corpus but must come from another source, e.g., human labeling. For instance, predicting the argument class (i.e., ARG0, ARG1, ...) of a constituent is an unobservable auxiliary problem (which is also the only usable unobservable auxiliary problem here), because the true outcomes (i.e., the argument classes) are only available from human labels annotated in NomBank.

For argument identification, we invent the following 20 binary unobservable auxiliary problems to take advantage of information previously unused at this stage:

> To predict the outcome of argument classification (i.e., ARG0, ARG1, ...) using the features of argument identification (*pred*, *subcat*, ...).

Thus for argument identification, we have 20 auxiliary problems (one auxiliary problem for predicting each of the argument classes ARG0, ARG1, ...) and one target problem (predicting whether a constituent is an argument) for the ASO algorithm described in Section 3.2.

In the argument classification task, the 20 binary target problems are also the unobservable auxiliary problems (one auxiliary problem for predicting each of the argument classes ARG0, ARG1, ...). Thus, we use the same 20 problems as both auxiliary problems and target problems.

We train an ASO classifier on sections 2 to 21 and test it on section 23. With the 20 unobservable auxiliary problems, we obtain the F1 scores reported in the last column of Table 3. In all the experiments, we keep $h = 20$, i.e., all the 20 columns of V_1 are kept.

Comparing the F1 score of ASO against that of the linear classifier in every task (i.e., identification, classification, combined, both auto parse configurations), the improvement achieved by ASO is statistically significant ($p < 0.05$) based on the χ^2 test. Comparing the F1 score of ASO against that of the maximum entropy classifier, the improvement in all but one task (argument classification) is statistically significant ($p < 0.05$). For argument classification, the improvement is not statistically significant ($p = 0.08$).

5.2 Observable auxiliary problems

Observable auxiliary problems are problems whose true outcome can be observed from a raw text corpus without additional externally provided labels. An example is to predict whether *hw=trader* from a constituent's other features, since the head word of a constituent can be obtained from the raw text alone. By definition, an observable auxiliary problem can always be formulated as predicting a feature of the training data. Depending on whether the baseline linear classifier already uses the feature to be predicted, we face two possibilities:

Predicting a used feature In auxiliary problems of this type, we must take care to remove the feature itself from the training data. For example, we must not use the feature *path* or *pred&path* to predict *path* itself.

Predicting an unused feature These auxiliary problems provide information that the classifier was previously unable to incorporate. The desirable characteristics of such a feature are:

1. The feature, although unused, should have been considered for the target problem so it is probably related to the target problem.

2. The feature should not be highly correlated with a used feature, e.g., since the *lastword* feature is used in argument identification, we will not consider predicting *lastword.pos* as an auxiliary problem.

Each chosen feature can create thousands of binary auxiliary problems. E.g., by choosing to predict *hw*, we can create auxiliary problems predicting whether *hw=to*, whether *hw=trader*, etc. To have more positive training samples, we only predict the most frequent features. Thus we will probably predict whether *hw=to*, but not whether *hw=trader*, since *to* occurs more frequently than *trader* as a head word.

5.2.1 Argument identification

In argument identification using gold parse trees, we experiment with predicting three unused features as auxiliary problems: *distance* (distance between the predicate and the constituent), *parent.lsis.hw* (head word of the parent constituent's left sister) and *parent.rsis.hw* (head word of the parent constituent's right sister). We then experiment with predicting four used features: *hw*, *lastword*, *ptype* and *path*.

The ASO classifier is trained on sections 2 to 21, and tested on section 23. Due to the large data size, we are unable to use more than 20 binary auxiliary problems or to experiment with combinations of them. The F1 scores are presented in Table 4.

5.2.2 Argument classification

In argument classification using gold parse trees and gold identification, we experiment with predicting three unused features *path*, *partialpath*, and *chunkseq* (concatenation of the phrase types of text chunks between the predicate and the constituent). We then experiment with predicting three used features *hw*, *lastword*, and *ptype*.

Combinations of these auxiliary problems are also tested. In *all combined*, we use the first 100 problems from each of the six groups of observable auxiliary problems. In *selected combined*, we use the first 100 problems from each of *path*, *chunkseq*, *lastword* and *ptype* problems.

The ASO classifier is trained on sections 2 to 21, and tested on section 23. The F1 scores are shown in Table 5.

feature to be predicted	F1
20 most frequent *distance*s	81.48
20 most frequent *parent.lsis.hw*s	81.51
20 most frequent *parent.rsis.hw*s	81.60
20 most frequent *hw*s	81.40
20 most frequent *lastword*s	81.33
20 most frequent *ptype*s	81.35
20 most frequent *path*s	81.47
linear baseline	81.34

Table 4: F1 scores of ASO with observable auxiliary problems on argument identification. All $h = 20$.

From Table 4 and 5, we observe that although the use of observable auxiliary problems consistently improves the performance of the classifier, the differences are small and not statistically significant. Further experiments combining unobservable and observable auxiliary problems fail to outperform ASO with unobservable auxiliary problems alone.

feature to be predicted	F1
300 most frequent *path*s	87.97
300 most frequent *partialpath*s	87.95
300 most frequent *chunkseq*s	88.09
300 most frequent *hw*s	87.93
300 most frequent *lastword*s	88.01
all 63 *ptype*s	88.05
all combined	87.95
selected combined	88.07
linear baseline	87.86

Table 5: F1 scores of ASO with observable auxiliary problems on argument classification. All $h = 100$.

In summary, our work shows that unobservable auxiliary problems significantly improve the performance of NomBank SRL. In contrast, observable auxiliary problems are not effective.

6 Discussions

Some of our experiments are limited by the extensive computing resources required for a fuller exploration. For instance, "predicting unused features" type of auxiliary problems might hold some hope for further improvement in argument identification, if a larger number of auxiliary problems can be used.

ASO has been demonstrated to be an effective semi-supervised learning algorithm (Ando and Zhang, 2005a; Ando and Zhang, 2005b; Ando, 2006). However, we have been unable to use unlabeled data to improve the accuracy. One possible reason is the cumulative noise from the many cascading steps involved in automatic SRL of unlabeled data: syntactic parse, predicate identification (where we identify nouns with at least one argument), argument identification, and finally argument classification, which reduces the effectiveness of adding unlabeled data using ASO.

7 Related work

Multi-output neural networks learn several tasks simultaneously. In addition to the target outputs,

(Caruana, 1997) discusses configurations where both used inputs and unused inputs (due to excessive noise) are utilized as additional outputs. In contrast, our work concerns linear predictors using empirical risk minimization.

A variety of auxiliary problems are tested in (Ando and Zhang, 2005a; Ando and Zhang, 2005b) in the semi-supervised settings, i.e., their auxiliary problems are generated from unlabeled data. This differs significantly from the supervised setting in our work, where only labeled data is used. While (Ando and Zhang, 2005b) uses "predicting used features" (previous/current/next word) as auxiliary problems with good results in named entity recognition, the use of similar observable auxiliary problems in our work gives no statistically significant improvements.

More recently, for the word sense disambiguation (WSD) task, (Ando, 2006) experimented with both supervised and semi-supervised auxiliary problems, although the auxiliary problems she used are different from ours.

8 Conclusion

In this paper, we have presented a novel application of Alternating Structure Optimization (ASO) to the Semantic Role Labeling (SRL) task on NomBank. The possible auxiliary problems are categorized and tested extensively. Our results outperform those reported in (Jiang and Ng, 2006). To the best of our knowledge, we achieve the highest SRL accuracy published to date on the English NomBank.

References

R. K. Ando and T. Zhang. 2005a. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*.

R. K. Ando and T. Zhang. 2005b. A high-performance semi-supervised learning method for text chunking. In *Proc. of ACL*.

R. K. Ando. 2006. Applying alternating structure optimization to word sense disambiguation. In *Proc. of CoNLL*.

C. F. Baker, C. J. Fillmore, and J. B. Lowe. 1998. The Berkeley FrameNet project. In *Proc. of COLING-ACL*.

S. Ben-David and R. Schuller. 2003. Exploiting task relatedness for multiple task learning. In *Proc. of COLT*.

X. Carreras and L. Marquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proc. of CoNLL*.

X. Carreras and L. Marquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proc. of CoNLL*.

R. Caruana. 1997. *Multitask Learning*. Ph.D. thesis, School of Computer Science, CMU.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proc. of ACL*.

T. Evgeniou and M. Pontil. 2004. Regularized multitask learning. In *Proc. of KDD*.

D. Gildea and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*.

Z. P. Jiang and H. T. Ng. 2006. Semantic role labeling of NomBank: A maximum entropy approach. In *Proc. of EMNLP*.

K. C. Litkowski. 2004. Senseval-3 task: automatic labeling of semantic roles. In *Proc. of SENSEVAL-3*.

A. Maurer. 2006. Bounds for linear multitask learning. *Journal of Machine Learning Research*.

A. Meyers, R. Reeves, C. Macleod, R. Szekeley, V. Zielinska, B. Young, and R. Grishman. 2004. The NomBank project: An interim report. In *Proc. of HLT/NAACL Workshop on Frontiers in Corpus Annotation*.

C. A. Micchelli and M. Pontil. 2005. Kernels for multitask learning. In *Proc. of NIPS*.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The Proposition Bank: an annotated corpus of semantic roles. *Computational Linguistics*.

S. S. Pradhan, H. Sun, W. Ward, J. H. Martin, and D. Jurafsky. 2004. Parsing arguments of nominalizations in English and Chinese. In *Proc. of HLT/NAACL*.

S. Pradhan, K. Hacioglu, V. Krugler, W. Ward, J. H. Martin, and D. Jurafsky. 2005. Support vector learning for semantic argument classification. *Machine Learning*.

K. Toutanova, A. Haghighi, and C. D. Manning. 2005. Joint learning improves semantic role labeling. In *Proc. of ACL*.

N. Xue and M. Palmer. 2004. Calibrating features for semantic role labeling. In *Proc. of EMNLP*.

N. Xue. 2006. Semantic role labeling of nominalized predicates in Chinese. In *Proc. of HLT/NAACL*.

A Simple, Similarity-based Model for Selectional Preferences

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

Abstract

We propose a new, simple model for the automatic induction of selectional preferences, using corpus-based semantic similarity metrics. Focusing on the task of semantic role labeling, we compute selectional preferences for semantic roles. In evaluations the similarity-based model shows lower error rates than both Resnik's WordNet-based model and the EM-based clustering model, but has coverage problems.

1 Introduction

Selectional preferences, which characterize typical arguments of predicates, are a very useful and versatile knowledge source. They have been used for example for syntactic disambiguation (Hindle and Rooth, 1993), word sense disambiguation (WSD) (McCarthy and Carroll, 2003) and semantic role labeling (SRL) (Gildea and Jurafsky, 2002).

The corpus-based induction of selectional preferences was first proposed by Resnik (1996). All later approaches have followed the same two-step procedure, first collecting argument headwords from a corpus, then generalizing to other, similar words. Some approaches have used WordNet for the generalization step (Resnik, 1996; Clark and Weir, 2001; Abe and Li, 1993), others EM-based clustering (Rooth et al., 1999).

In this paper we propose a new, simple model for selectional preference induction that uses corpus-based semantic similarity metrics, such as Cosine or Lin's (1998) mutual information-based metric, for the generalization step. This model does not require any manually created lexical resources. In addition, the corpus for computing the similarity metrics can be freely chosen, allowing greater variation in the domain of generalization than a fixed lexical resource.

We focus on one application of selectional preferences: semantic role labeling. The argument positions for which we compute selectional preferences will be semantic roles in the FrameNet (Baker et al., 1998) paradigm, and the predicates we consider will be semantic classes of words rather than individual words (which means that different preferences will be learned for different senses of a predicate word). In SRL, the two most pressing issues today are (1) the development of strong semantic features to complement the current mostly syntactically-based systems, and (2) the problem of the domain dependence (Carreras and Marquez, 2005). In the CoNLL-05 shared task, participating systems showed about 10 points F-score difference between in-domain and out-of-domain test data. Concerning (1), we focus on selectional preferences as the strongest candidate for informative semantic features. Concerning (2), the corpus-based similarity metrics that we use for selectional preference induction open up interesting possibilities of mixing domains.

We evaluate the similarity-based model against Resnik's WordNet-based model as well as the EM-based clustering approach. In the evaluation, the similarity-model shows lower error rates than both Resnik's WordNet-based model and the EM-based clustering model. However, the EM-based clustering model has higher coverage than both other paradigms.

Plan of the paper. After discussing previ-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 216–223,
Prague, Czech Republic, June 2007.

ous approaches to selectional preference induction in Section 2, we introduce the similarity-based model in Section 3. Section 4 describes the data used for the experiments reported in Section 5, and Section 6 concludes.

2 Related Work

Selectional restrictions and selectional preferences that predicates impose on their arguments have long been used in semantic theories, (see e.g. (Katz and Fodor, 1963; Wilks, 1975)). The induction of selectional preferences from corpus data was pioneered by Resnik (1996). All subsequent approaches have followed the same two-step procedure, first collecting argument headwords from a corpus, then generalizing over the seen headwords to similar words. Resnik uses the WordNet noun hierarchy for generalization. His information-theoretic approach models the *selectional preference strength* of an argument position[1] r_p of a predicate p as

$$S(r_p) = \sum_c P(c|r_p) \log \frac{P(c|r_p)}{P(c)}$$

where the c are WordNet synsets. The preference that r_p has for a given synset c_0, the *selectional association* between the two, is then defined as the contribution of c_0 to r_p's selectional preference strength:

$$A(r_p, c_0) = \frac{P(c_0|r_p) \log \frac{P(c_0|r_p)}{P(c_0)}}{S(r_p)}$$

Further WordNet-based approaches to selectional preference induction include Clark and Weir (2001), and Abe and Li (1993). Brockmann and Lapata (2003) perform a comparison of WordNet-based models.

Rooth et al. (1999) generalize over seen headwords using EM-based clustering rather than WordNet. They model the probability of a word w occurring as the argument r_p of a predicate p as being independently conditioned on a set of classes C:

$$P(r_p, w) = \sum_{c \in C} P(c, r_p, w) = \sum_{c \in C} P(c) P(r_p|c) P(w|c)$$

The parameters $P(c)$, $P(r_p|c)$ and $P(w|c)$ are estimated using the EM algorithm.

While there have been no isolated comparisons of the two generalization paradigms that we are aware of, Gildea and Jurafsky's (2002) task-based evaluation has found clustering-based approaches to have better coverage than WordNet generalization, that is, for a given role there are more words for which they can state a preference.

3 Model

The approach we are proposing makes use of two corpora, a **primary corpus** and a **generalization corpus** (which may, but need not, be identical). The primary corpus is used to extract tuples (p, r_p, w) of a **predicate**, an **argument position** and a **seen headword**. The generalization corpus is used to compute a corpus-based semantic similarity metric.

Let $\text{Seen}(r_p)$ be the set of seen headwords for an argument r_p of a predicate p. Then we model the selectional preference S of r_p for a possible headword w_0 as a weighted sum of the similarities between w_0 and the seen headwords:

$$S_{r_p}(w_0) = \sum_{w \in \text{Seen}(r_p)} \text{sim}(w_0, w) \cdot \text{wt}_{r_p}(w)$$

$\text{sim}(w_0, w)$ is the similarity between the seen and the potential headword, and $\text{wt}_{r_p}(w)$ is the weight of seen headword w.

Similarity $\text{sim}(w_0, w)$ will be computed on the generalization corpus, again on the basis of extracted tuples (p, r_p, w). We will be using the similarity metrics shown in Table 1: Cosine, the Dice and Jaccard coefficients, and Hindle's (1990) and Lin's (1998) mutual information-based metrics. We write f for frequency, I for mutual information, and $R(w)$ for the set of arguments r_p for which w occurs as a headword.

In this paper we only study corpus-based metrics. The sim function can equally well be instantiated with a WordNet-based metric (for an overview see Budanitsky and Hirst (2006)), but we restrict our experiments to corpus-based metrics (a) in the interest of greatest possible

[1] We write r_p to indicate predicate-specific roles, like "the direct object of catch", rather than just "obj".

$$\mathrm{sim}_{\mathrm{cosine}}(w,w') = \frac{\sum_{r_p} f(w,r_p)\cdot f(w',r_p)}{\sqrt{\sum_{r_p} f(w,r_p)^2}\cdot\sqrt{\sum_{r_p} f(w',r_p)^2}} \qquad \mathrm{sim}_{\mathrm{Dice}}(w,w') = \frac{2\cdot|R(w)\cap R(w')|}{|R(w)|+|R(w')|}$$

$$\mathrm{sim}_{\mathrm{Lin}}(w,w') = \frac{\sum_{r_p\in R(w)\cap R(w')} I(w,r,p)I(w',r,p)}{\sum_{r_p\in R(w)} I(w,r,p)\sum_{r_p\in R(w)} I(w',r,p)} \qquad \mathrm{sim}_{\mathrm{Jaccard}}(w,w') = \frac{|R(w)\cap R(w')|}{|R(w)\cup R(w')|}$$

$$\mathrm{sim}_{\mathrm{Hindle}}(w,w') = \sum_{r_p} \mathrm{sim}_{\mathrm{Hindle}}(w,w',r_p) \qquad \text{where}$$

$$\mathrm{sim}_{\mathrm{Hindle}}(w,w',r_p) = \begin{cases} \min(I(w,r_p),I(w',r_p)) & \text{if } I(w,r_p)>0 \text{ and } I(w',r_p)>0 \\ \mathrm{abs}(\max(I(w,r_p),I(w',r_p))) & \text{if } I(w,r_p)<0 \text{ and } I(w',r_p)<0 \\ 0 & \text{else} \end{cases}$$

Table 1: Similarity measures used

resource-independence and (b) in order to be able to shape the similarity metric by the choice of generalization corpus.

For the headword weights $\mathrm{wt}_{r_p}(w)$, the simplest possibility is to assume a uniform weight distribution, i.e. $\mathrm{wt}_{r_p}(w) = 1$. In addition, we test a frequency-based weight, i.e. $\mathrm{wt}_{r_p}(w) = f(w,r_p)$, and inverse document frequency, which weighs a word according to its discriminativity: $\mathrm{wt}_{r_p}(w) = \log \frac{\text{num. words}}{\text{num. words to whose context w belongs}}$.

This similarity-based model of selectional preferences is a straightforward implementation of the idea of generalization from seen headwords to other, similar words. Like the clustering-based model, it is not tied to the availability of WordNet or any other manually created resource. The model uses two corpora, a primary corpus for the extraction of seen headwords and a generalization corpus for the computation of semantic similarity metrics. This gives the model flexibility to influence the similarity metric through the choice of text domain of the generalization corpus.

Instantiation used in this paper. Our aim is to compute selectional preferences for semantic roles. So we choose a particular instantiation of the similarity-based model that makes use of the fact that the two-corpora approach allows us to use different notions of "predicate" and "argument" in the primary and generalization corpus. Our primary corpus will consist of manually semantically annotated data, and we will use semantic verb classes as predicates and semantic roles as arguments. Examples of extracted (p, r_p, w) tuples are (Morality_evaluation, Evaluee, gamblers) and (Placing, Goal, briefcase). Semantic similarity, on the other hand, will be computed on automatically syntactically parsed corpus, where the predicates are words and the arguments are syntactic dependents. Examples of extracted (p, r_p, w) tuples from the generalization corpus include (catch, obj, frogs) and (intervene, in, deal).[2]

This instantiation of the similarity-based model allows us to compute word sense specific selectional preferences, generalizing over manually semantically annotated data using automatically syntactically annotated data.

4 Data

We use FrameNet (Baker et al., 1998), a semantic lexicon for English that groups words in semantic classes called *frames* and lists semantic roles for each frame. The FrameNet 1.3 annotated data comprises 139,439 sentences from the British National Corpus (BNC). For our experiments, we chose 100 frame-specific semantic roles at random, 20 each from five frequency bands: 50-100 annotated occurrences of the role, 100-200 occurrences, 200-500, 500-1000, and more than 1000 occurrences. The annotated data for these 100 roles comprised 59,608 sentences, our primary corpus. To determine headwords of the semantic roles, the corpus was parsed using the Collins (1997) parser.

Our generalization corpus is the BNC. It was parsed using Minipar (Lin, 1993), which is considerably faster than the Collins parser but failed to parse about a third of all sentences.

[2]For details about the syntactic and semantic analyses used, see Section 4.

Accordingly, the arguments r extracted from the generalization corpus are Minipar dependencies, except that paths through preposition nodes were collapsed, using the preposition as the dependency relation. We obtained parses for 5,941,811 sentences of the generalization corpus.

The EM-based clustering model was computed with all of the FrameNet 1.3 data (139,439 sentences) as input. Resnik's model was trained on the primary corpus (59,608 sentences).

5 Experiments

In this section we describe experiments comparing the similarity-based model for selectional preferences to Resnik's WordNet-based model and to an EM-based clustering model[3]. For the similarity-based model we test the five similarity metrics and three weighting schemes listed in section 3.

Experimental design

Like Rooth et al. (1999) we evaluate selectional preference induction approaches in a pseudo-disambiguation task. In a test set of pairs (r_p, w), each headword w is paired with a confounder w' chosen randomly from the BNC according to its frequency[4]. Noun headwords are paired with noun confounders in order not to disadvantage Resnik's model, which only works with nouns. The headword/confounder pairs are only computed once and reused in all cross-validation runs. The task is to choose the more likely role headword from the pair (w, w').

In the main part of the experiment, we count a pair as *covered* if both w and w' are assigned some level of preference by a model (*"full coverage"*). We contrast this with another condition, where we count a pair as covered if at least one of the two words w, w' is assigned a level of preference by a model (*"half coverage"*). If only one is assigned a preference, that word is counted as chosen.

To test the performance difference between models for significance, we use Dietterich's 5x2cv (Dietterich, 1998). The test involves five 2-fold cross-validation runs. Let $d_{i,j}$ ($i \in \{1,2\}, j \in \{1,\ldots,5\}$) be the difference in error rates between the two models when using split i of cross-validation run j as training data. Let $s_j^2 = (d_{1,j} - \bar{d}_j)^2 + (d_{2,j} - \bar{d}_j)^2$ be the variance for cross-validation run j, with $\bar{d}_j = \frac{d_{1,j}+d_{2,j}}{2}$. Then the 5x2cv $\tilde{t}$ statistic is defined as

$$\tilde{t} = \frac{d_{1,1}}{\sqrt{\frac{1}{5}\sum_{j=1}^{5} s_j^2}}$$

Under the null hypothesis, the $\tilde{t}$ statistic has approximately a t distribution with 5 degrees of freedom.[5]

	Error Rate	Coverage
Cosine	0.2667	0.3284
Dice	0.1951	0.3506
Hindle	0.2059	0.3530
Jaccard	**0.1858**	0.3506
Lin	**0.1635**	0.2214
EM 30/20	0.3115	0.5460
EM 40/20	0.3470	**0.9846**
Resnik	0.3953	0.3084

Table 2: Error rate and coverage (micro-average), similarity-based models with uniform weights.

Results and discussion

Error rates. Table 2 shows error rates and coverage for the different selectional preference induction methods. The first five models are similarity-based, computed with uniform weights. The name in the first column is the name of the similarity metric used. Next come EM-based clustering models, using 30 (40) clusters and 20 re-estimation steps[6], and the last row lists the results for Resnik's WordNet-based method. Results are micro-averaged.

The table shows very low error rates for the similarity-based models, up to 15 points lower than the EM-based models. The error rates

[3]We are grateful to Carsten Brockmann and Detlef Prescher for the use of their software.

[4]We exclude potential confounders that occur less than 30 or more than 3,000 times.

[5]Since the 5x2cv test fails when the error rates vary wildly, we excluded cases where error rates differ by 0.8 or more across the 10 runs, using the threshold recommended by Dietterich.

[6]The EM-based clustering software determines good values for these two parameters through pseudo-disambiguation tests on the training data.

	Cos	Dic	Hin	Jac	Lin	EM 40/20	Resnik
Cos		-16 *(73)*	-12 *(73)*	-18 *(74)*	-22 *(57)*	11 *(67)*	11 *(74)*
Dic	16 *(73)*		2 *(74)*	-8 *(85)*	-10 *(64)*	39 *(47)*	27 *(62)*
Hin	12 *(73)*	-2 *(74)*		-8 *(75)*	-11 *(63)*	33 *(57)*	16 *(67)*
Jac	18 *(74)*	8 *(85)*	8 *(75)*		-7 *(68)*	42 *(45)*	30 *(62)*
Lin	22 *(57)*	10 *(64)*	11 *(63)*	7 *(68)*		29 *(41)*	28 *(51)*
EM 40/20	-11 *(67)*	-39 *(47)*	-33 *(57)*	-42 *(45)*	-29 *(41)*		3 *(72)*
Resnik	-11 *(74)*	-27 *(62)*	-16 *(67)*	-30 *(62)*	-28 *(51)*	-3 *(72)*	

Table 3: Comparing similarity measures: number of wins minus losses (in brackets non-significant cases) using Dietterich's 5x2cv; uniform weights; condition (1): both members of a pair must be covered

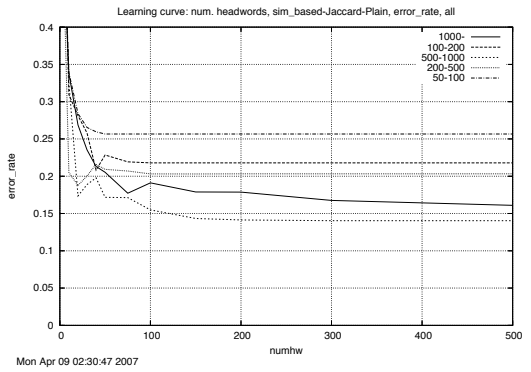


Figure 1: Learning curve: seen headwords versus error rate by frequency band, Jaccard, uniform weights

	50-100	100-200	200-500	500-1000	1000-
Cos	0.3167	0.3203	0.2700	0.2534	0.2606
Jac	0.1802	0.2040	0.1761	0.1706	0.1927

Table 4: Error rates for similarity-based models, by semantic role frequency band. Micro-averages, uniform weights

of Resnik's model are considerably higher than both the EM-based and the similarity-based models, which is unexpected. While EM-based models have been shown to work better in SRL tasks (Gildea and Jurafsky, 2002), this has been attributed to the difference in coverage.

In addition to the *full coverage* condition, we also computed error rate and coverage for the *half coverage* case. In this condition, the error rates of the EM-based models are unchanged, while the error rates for all similarity-based models as well as Resnik's model rise to values between 0.4 and 0.6. So the EM-based model tends to have preferences only for the "right" words. Why this is so is not clear. It may be a genuine property, or an artifact of the FrameNet data, which only contains chosen, illustrative sentences for each frame. It is possible that these sentences have fewer occurrences of highly frequent but semantically less informative role headwords like "it" or "that" exactly because of their illustrative purpose.

Table 3 inspects differences between error rates using Dietterich's 5x2cv, basically confirming Table 2. Each cell shows the wins minus losses for the method listed in the row when compared against the method in the column. The number of cases that did not reach significance is given in brackets.

Coverage. The coverage rates of the similarity-based models, while comparable to Resnik's model, are considerably lower than for EM-based clustering, which achieves good coverage with 30 and almost perfect coverage with 40 clusters (Table 2). While peculiarities of the FrameNet data may have influenced the results in the EM-based model's favor (see the discussion of the *half coverage* condition above), the low coverage of the similarity-based models is still surprising. After all, the generalization corpus of the similarity-based models is far larger than the corpus used for clustering. Given the learning curve in Figure 1 it is unlikely that the reason for the lower coverage is data sparseness. However, EM-based clustering is a soft clustering method, which relates every predicate and every headword to every cluster, if only with a very low probabil-

ity. In similarity-based models, on the other hand, two words that have never been seen in the same argument slot in the generalization corpus will have zero similarity. That is, a similarity-based model can assign a level of preference for an argument r_p and word w_0 only if $R(w_0) \cap R(\text{Seen}(r_p))$ is nonempty. Since the flexibility of similarity-based models extends to the vector space for computing similarities, one obvious remedy to the coverage problem would be the use of a less sparse vector space. Given the low error rates of similarity-based models, it may even be advisable to use two vector spaces, backing off to the denser one for words not covered by the sparse but highly accurate space used in this paper.

Parameters of similarity-based models. Besides the similarity metric itself, which we discuss below, parameters of the similarity-based models include the number of seen headwords, the weighting scheme, and the number of similar words for each headword.

Table 4 breaks down error rates by semantic role frequency band for two of the similarity-based models, micro-averaging over roles of the same frequency band and over cross-validation runs. As the table shows, there was some variation across frequency bands, but not as much as between models.

The question of the number of seen headwords necessary to compute selectional preferences is further explored in Figure 1. The figure charts the number of seen headwords against error rate for a Jaccard similarity-based model (uniform weights). As can be seen, error rates reach a plateau at about 25 seen headwords for Jaccard. For other similarity metrics the result is similar.

The weighting schemes wt_{r_p} had surprisingly little influence on results. For Jaccard similarity, the model had an error rate of 0.1858 for uniform weights, 0.1874 for frequency weighting, and 0.1806 for discriminativity. For other similarity metrics the results were similar.

A cutoff was used in the similarity-based model: For each seen headword, only the 500 most similar words (according to a given similarity measure) were included in the computation; for all others, a similarity of 0 was assumed. Experiments testing a range of values for this parameter show that error rates stay stable for parameter values ≥ 200.

	Cos	Dic	Hin	Jac	Lin
(a) Freq. sim.	1889	3167	2959	3167	860
(b) Freq. wins	65%	73%	79%	72%	58%
(c) Num. sim.	81	60	67	60	66
(d) Intersec.	7.3	2.3	7.2	2.1	0.5

Table 5: Comparing sim. metrics: (a) avg. freq. of similar words; (b) % of times the more frequent word won; (c) number of distinct similar words per seen headword; (d) avg. size of intersection between roles

So similarity-based models seem not overly sensitive to the weighting scheme used, the number of seen headwords, or the number of similar words per seen headword. The difference between similarity metrics, however, is striking.

Differences between similarity metrics. As Table 2 shows, Lin and Jaccard worked best (though Lin has very low coverage), Dice and Hindle not as good, and Cosine showed the worst performance. To determine possible reasons for the difference, Table 5 explores properties of the five similarity measures.

Given a set $S = \text{Seen}(r_p)$ of seen headwords for some role r_p, each similarity metric produces a set $\text{like}(S)$ of words that have nonzero similarity to S, that is, to at least one word in S. Line (a) shows the average frequency of words in $\text{like}(S)$. The results confirm that the Lin and Cosine metrics tend to propose less frequent words as similar.

Line (b) pursues the question of the frequency bias further, showing the percentage of headword/confounder pairs for which the more frequent of the two words "won" in the pseudo-disambiguation task (using uniform weights). This it is an indirect estimate of the frequency bias of a similarity metric. Note that the headword actually was more frequent than the confounder in only 36% of all pairs.

These first two tests do not yield any explanation for the low performance of Cosine, as the results they show do not separate Cosine from

Jaccard	Cosine
Ride_vehicle:Vehicle truck 0.05 boat 0.05 coach 0.04 van 0.04 ship 0.04 lorry 0.04 creature 0.04 flight 0.04 guy 0.04 carriage 0.04 helicopter 0.04 lad 0.04	**Ride_vehicle:Vehicle** it 1.18 there 0.88 they 0.43 that 0.34 i 0.23 ship 0.19 second one 0.19 machine 0.19 e 0.19 other one 0.19 response 0.19 second 0.19
Ingest_substance:Substance loaf 0.04 ice cream 0.03 you 0.03 some 0.03 that 0.03 er 0.03 photo 0.03 kind 0.03 he 0.03 type 0.03 thing 0.03 milk 0.03	**Ingest_substance:Substance** there 1.23 that 0.50 object 0.27 argument 0.27 theme 0.27 version 0.27 machine 0.26 result 0.26 response 0.25 item 0.25 concept 0.25 s 0.24

Table 6: Highest-ranked induced headwords (seen headwords omitted) for two semantic classes of the verb "take": similarity-based models, Jaccard and Cosine, uniform weights.

all other metrics. Lines (c) and (d), however, do just that. Line (c) looks at the size of like(S). Since we are using a cutoff of 500 similar words computed per word in S, the size of like(S) can only vary if the same word is suggested as similar for several seen headwords in S. This way, the size of like(S) functions as an indicator of the degree of uniformity or similarity that a similarity metric "perceives" among the members of S. To facilitate comparison across frequency bands, line (c) normalizes by the size of S, showing $\frac{|\text{like}(S)|}{|S|}$ micro-averaged over all roles. Here we see that Cosine seems to "perceive" considerably less similarity among the seen headwords than any of the other metrics.

Line (d) looks at the sets $s_{25}(r)$ of the 25 most preferred potential headwords of roles r, showing the average size of the intersection $s_{25}(r) \cap s_{25}(r')$ between two roles (preferences computed with uniform weights). It indicates another possible reason for Cosine's problem: Cosine seems to keep proposing the same words as similar for different roles. We will see this tendency also in the sample results we discuss next.

Sample results. Table 6 shows samples of headwords induced by the similarity-based model for two FrameNet senses of the verb "take": Ride_vehicle ("take the bus") and Ingest_substance ("take drugs"), a semantic class that is exclusively about ingesting controlled substances. The semantic role Vehicle of the Ride_vehicle frame and the role Substance of Ingest_substance are both typically realized as the direct object of "take". The table only shows new induced headwords; seen headwords were omitted from the list.

The particular implementation of the similarity-based model we have chosen, using frames and roles as predicates and arguments in the primary corpus, should enable the model to compute preferences specific to word senses. The sample in Table 6 shows that this is indeed the case: The preferences differ considerably for the two senses (frames) of "take", at least for the Jaccard metric, which shows a clear preference for vehicles for the Vehicle role. The Substance role of Ingest_substance is harder to characterize, with very diverse seen headwords such as "crack", "lines", "fluid", "speed". While the highest-ranked induced words for Jaccard do include three food items, there is no word, with the possible exception of "ice cream", that could be construed as a controlled substance. The induced headwords for the Cosine metric are considerably less pertinent for both roles and show the above-mentioned tendency to repeat some high-frequency words.

The inspection of "take" anecdotally confirms that different selectional preferences are learned for different senses. This point (which comes down to the usability of selectional preferences for WSD) should be verified in an empirical evaluation, possibly in another pseudo-disambiguation task, choosing as confounders seen headwords for *other* senses of a predicate word.

6 Conclusion

We have introduced the similarity-based model for inducing selectional preferences. Computing selectional preference as a weighted sum of similarities to seen headwords, it is a straight-

forward implementation of the idea of generalization from seen headwords to other, similar words. The similarity-based model is particularly simple and easy to compute, and seems not very sensitive to parameters. Like the EM-based clustering model, it is not dependent on lexical resources. It is, however, more flexible in that it induces similarities from a separate generalization corpus, which allows us to control the similarities we compute by the choice of text domain for the generalization corpus. In this paper we have used the model to compute sense-specific selectional preferences for semantic roles.

In a pseudo-disambiguation task the similarity-based model showed error rates down to 0.16, far lower than both EM-based clustering and Resnik's WordNet model. However its coverage is considerably lower than that of EM-based clustering, comparable to Resnik's model. The most probable reason for this is the sparsity of the underlying vector space. The choice of similarity metric is critical in similarity-based models, with Jaccard and Lin achieving the best performance, and Cosine surprisingly bringing up the rear.

Next steps will be to test the similarity-based model "in vivo", in an SRL task; to test the model in a WSD task; to evaluate the model on a primary corpus that is not semantically analyzed, for greater comparability to previous approaches; to explore other vector spaces to address the coverage issue; and to experiment on domain transfer, using an appropriate generalization corpus to induce selectional preferences for a domain different from that of the primary corpus. This is especially relevant in view of the domain-dependence problem that SRL faces.

Acknowledgements Many thanks to Jason Baldridge, Razvan Bunescu, Stefan Evert, Ray Mooney, Ulrike and Sebastian Padó, and Sabine Schulte im Walde for helpful discussions.

References

N. Abe and H. Li. 1993. Learning word association norms using tree cut pair models. In *Proceedings of ICML 1993*.

C. Baker, C. Fillmore, and J. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of COLING-ACL 1998*, Montreal, Canada.

C. Brockmann and M. Lapata. 2003. Evaluating and combining approaches to selectional preference acquisition. In *Proceedings of EACL 2003*, Budapest.

A. Budanitsky and G. Hirst. 2006. Evaluating WordNet-based measures of semantic distance. *Computational Linguistics*, 32(1).

X. Carreras and L. Marquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL-05*, Ann Arbor, MI.

S. Clark and D. Weir. 2001. Class-based probability estimation using a semantic hierarchy. In *Proceedings of NAACL 2001*, Pittsburgh, PA.

M. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL 1997*, Madrid, Spain.

T. Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10:1895–1923.

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

D. Hindle and M. Rooth. 1993. Structural ambiguity and lexical relations. *Computational Linguistics*, 19(1).

D. Hindle. 1990. Noun classification from predicate-argument structures. In *Proceedings of ACL 1990*, Pittsburg, Pennsylvania.

J. Katz and J. Fodor. 1963. The structure of a semantic theory. *Language*, 39(2).

D. Lin. 1993. Principle-based parsing without overgeneration. In *Proceedings of ACL 1993*, Columbus, OH.

D. Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL 1998*, Montreal, Canada.

D. McCarthy and J. Carroll. 2003. Disambiguating nouns, verbs and adjectives using automatically acquired selectional preferences. *Computatinal Linguistics*, 29(4).

P. Resnik. 1996. Selectional constraints: An information-theoretic model and its computational realization. *Cognition*, 61:127–159.

M. Rooth, S. Riezler, D. Prescher, G. Carroll, and F. Beil. 1999. Inducing an semantically annotated lexicon via EM-based clustering. In *Proceedings of ACL 1999*, Maryland.

Y. Wilks. 1975. Preference semantics. In E. Keenan, editor, *Formal Semantics of Natural Language*. Cambridge University Press.

SVM Model Tampering and Anchored Learning: A Case Study in Hebrew NP Chunking

Yoav Goldberg and Michael Elhadad
Computer Science Department
Ben Gurion University of the Negev
P.O.B 653 Be'er Sheva 84105, Israel
yoavg,elhadad@cs.bgu.ac.il

Abstract

We study the issue of porting a known NLP method to a language with little existing NLP resources, specifically Hebrew SVM-based chunking. We introduce two SVM-based methods – Model Tampering and Anchored Learning. These allow fine grained analysis of the learned SVM models, which provides guidance to identify errors in the training corpus, distinguish the role and interaction of lexical features and eventually construct a model with ∼10% error reduction. The resulting chunker is shown to be robust in the presence of noise in the training corpus, relies on less lexical features than was previously understood and achieves an F-measure performance of 92.2 on automatically PoS-tagged text. The SVM analysis methods also provide general insight on SVM-based chunking.

1 Introduction

While high-quality NLP corpora and tools are available in English, such resources are difficult to obtain in most other languages. Three challenges must be met when adapting results established in English to another language: (1) acquiring high quality annotated data; (2) adapting the English task definition to the nature of a different language, and (3) adapting the algorithm to the new language. This paper presents a case study in the adaptation of a well known task to a language with few NLP resources available. Specifically, we deal with SVM based Hebrew NP chunking. In (Goldberg et al., 2006), we established that the task is not trivially transferable to Hebrew, but reported that SVM based chunking (Kudo and Matsumoto, 2000) performs well. We extend that work and study the problem from 3 angles: (1) how to deal with a corpus that is smaller and with a higher level of noise than is available in English; we propose techniques that help identify 'suspicious' data points in the corpus, and identify how robust the model is in the presence of noise; (2) we compare the task definition in English and in Hebrew through quantitative evaluation of the differences between the two languages by analyzing the relative importance of features in the learned SVM models; and (3) we analyze the structure of learned SVM models to better understand the characteristics of the chunking problem in Hebrew.

While most work on chunking with machine learning techniques tend to treat the classification engine as a black-box, we try to investigate the resulting classification model in order to understand its inner working, strengths and weaknesses. We introduce two SVM-based methods – Model Tampering and Anchored Learning – and demonstrate how a fine-grained analysis of SVM models provides insights on all three accounts. The understanding of the relative contribution of each feature in the model helps us construct a better model, which achieves ∼10% error reduction in Hebrew chunking, as well as identify corpus errors. The methods also provide general insight on SVM-based chunking.

2 Previous Work

NP chunking is the task of marking the boundaries of simple noun-phrases in text. It is a well studied problem in English, and was the focus of CoNLL2000's Shared Task (Sang and Buchholz,

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 224–231,
Prague, Czech Republic, June 2007.

2000). Early attempts at NP Chunking were rule learning systems, such as the Error Driven Pruning method of Pierce and Cardie (1998). Following Ramshaw and Marcus (1995), the current dominant approach is formulating chunking as a classification task, in which each word is classified as the (B)eginning, (I)nside or (O)outside of a chunk. Features for this classification usually involve local context features. Kudo and Matsumoto (2000) used SVM as a classification engine and achieved an F-Score of 93.79 on the shared task NPs. Since SVM is a binary classifier, to use it for the 3-class classification of the chunking task, 3 different classifiers {B/I, B/O, I/O} were trained and their majority vote was taken.

NP chunks in the shared task data are BaseNPs, which are non-recursive NPs, a definition first proposed by Ramshaw and Marcus (1995). This definition yields good NP chunks for English. In (Goldberg et al., 2006) we argued that it is not applicable to Hebrew, mainly because of the prevalence of the Hebrew's construct state ($smixut$). $Smixut$ is similar to a noun-compound construct, but one that can join a noun (with a special morphological marking) with a full NP. It appears in about 40% of Hebrew NPs. We proposed an alternative definition (termed SimpleNP) for Hebrew NP chunks. A SimpleNP cannot contain embedded relatives, prepositions, VPs and NP-conjunctions (except when they are licensed by $smixut$). It can contain $smixut$, possessives (even when they are attached by the 'של/of' preposition) and partitives (and, therefore, allows for a limited amount of recursion). We applied this definition to the Hebrew Tree Bank (Sima'an et al., 2001), and constructed a moderate size corpus (about 5,000 sentences) for Hebrew SimpleNP chunking. SimpleNPs are different than English BaseNPs, and indeed some methods that work well for English performed poorly on Hebrew data. However, we found that chunking with SVM provides good result for Hebrew SimpleNPs. We analyzed that this success comes from SVM's ability to use lexical features, as well as two Hebrew morphological features, namely "number" and "construct-state".

One of the main issues when dealing with Hebrew chunking is that the available tree bank is rather small, and since it is quite new, and has not been used intensively, it contains a certain amount of inconsistencies and tagging errors. In addition, the identification of SimpleNPs from the tree bank also introduces some errors. Finally, we want to investigate chunking in a scenario where PoS tags are assigned automatically and chunks are then computed. The Hebrew PoS tagger we use introduces about 8% errors (compared with about 4% in English). We are, therefore, interested in identifying errors in the chunking corpus, and investigating how the chunker operates in the presence of noise in the PoS tag sequence.

3 Model Tampering

3.1 Notation and Technical Review

This section presents notation as well as a technical review of SVM chunking details relevant to the current study. Further details can be found in Kudo and Matsumoto (2000; 2003).

SVM (Vapnik, 1995) is a supervised binary classifier. The input to the learner is a set of l training samples $(x_1, y_1), \ldots, (x_l, y_l)$, $x \in R^n$, $y \in \{+1, -1\}$. x_i is an n dimensional feature vector representing the ith sample, and y_i is the label for that sample. The result of the learning process is the set SV of Support Vectors, the associated weights α_i, and a constant b. The Support Vectors are a subset of the training vectors, and together with the weights and b they define a hyperplane that optimally separates the training samples. The basic SVM formulation is of a linear classifier, but by introducing a kernel function K that non-linearly transforms the data from R^n into a higher dimensional space, SVM can be used to perform non-linear classification. SVM's decision function is: $y(x) = sgn\left(\sum_{j \in SV} y_j \alpha_j K(x_j, x) + b\right)$ where x is an n dimensional feature vector to be classified. In the linear case, K is a dot product operation and the sum $w = \sum y_j \alpha_j x_j$ is an n dimensional weight vector assigning weight for each of the n features. The other kernel function we consider in this paper is a polynomial kernel of degree 2: $K(x_i, x_j) = (x_i \cdot x_j + 1)^2$. When using binary valued features, this kernel function essentially implies that the classifier considers not only the explicitly specified features, but also all available pairs of features. In order to cope with inseparable data, the learning process of SVM allows for some misclassification, the amount of which is determined by a

parameter C, which can be thought of as a penalty for each misclassified training sample.

In SVM based chunking, each word and its context is considered a learning sample. We refer to the word being classified as w_0, and to its part-of-speech (PoS) tag, morphology, and B/I/O tag as p_0, m_0 and t_0 respectively. The information considered for classification is $w_{-cw} \ldots w_{cw}$, $p_{-cp} \ldots p_{cp}$, $m_{-cm} \ldots m_{cm}$ and $t_{-ct} \ldots t_{-1}$. The feature vector F is an indexed list of all the features present in the corpus. A feature f_i of the form $w_{+1} = dog$ means that the word following the one being classified is 'dog'. Every learning sample is represented by an $n = |F|$ dimensional binary vector x. $x_i = 1$ iff the feature f_i is active in the given sample, and 0 otherwise. This encoding leads to extremely high dimensional vectors, due to the lexical features $w_{-cw} \ldots w_{cw}$.

3.2 Introducing Model Tampering

An important observation about SVM classifiers is that features which are not active in any of the Support Vectors have no effect on the classifier decision. We introduce Model Tampering, a procedure in which we change the Support Vectors in a model by forcing some values in the vectors to 0.

The result of this procedure is a new Model in which the deleted features never take part in the classification.

Model tampering is different than feature selection: on the one hand, it is a method that helps us identify irrelevant features in a model after training; on the other hand, and this is the key insight, removing features **after** training is not the same as removing them before training. The presence of the low-relevance features during training has an impact on the generalization performed by the learner as shown below.

3.3 The Role of Lexical Features

In Goldberg *et al.* (2006), we have established that using lexical features increases the chunking F-measure from 78 to over 92 on the Hebrew Treebank. We refine this observation by using Model Tampering, in order to assess the importance of lexical features in NP Chunking. We are interested in identifying which specific lexical items and contexts impact the chunking decision, and quantifying their effect. Our method is to train a chunking model on a given training corpus, tamper with the resulting model in various ways and measure the performance[1] of the tampered models on a test corpus.

3.4 Experimental Setting

We conducted experiments both for English and Hebrew chunking. For the Hebrew experiments, we use the corpora of (Goldberg et al., 2006). The first one is derived from the original Treebank by projecting the full syntactic tree, constructed manually, onto a set of NP chunks according to the SimpleNP rules. We refer to the resulting corpus as HEB_{Gold} since PoS tags are fully reliable. The HEB_{Err} version of the corpus is obtained by projecting the chunk boundaries on the sequence of PoS and morphology tags obtained by the automatic PoS tagger of Adler & Elhadad (2006). This corpus includes an error rate of about 8% on PoS tags. The first 500 sentences are used for testing, and the rest for training. The corpus contains 27K NP chunks. For the English experiments, we use the now-standard training and test sets that were introduced in (Marcus and Ramshaw, 1995)[2]. Training was done using Kudo's YAMCHA toolkit[3]. Both Hebrew and English models were trained using a polynomial kernel of degree 2, with $C = 1$. For English, the features used were: $w_{-2} \ldots w_2$, $p_{-2} \ldots p_2$, $t_{-2} \ldots t_{-1}$. The same features were used for Hebrew, with the addition of $m_{-2} \ldots m_2$. These are the same settings as in (Kudo and Matsumoto, 2000; Goldberg et al., 2006).

3.5 Tamperings

We experimented with the following tamperings:

TopN – We define *model feature count* to be the number of Support Vectors in which a feature is active in a given classifier. This tampering leaves in the model only the top N lexical features in each classifier, according to their count.

NoPOS – all the lexical features corresponding to a given part-of-speech are removed from the model. For example, in a NoJJ tampering, all the features of the form $w_i = X$ are removed from all the support vectors in which $p_i = JJ$ is active.

Loc≠i – all the lexical features with index i are removed from the model *e.g.*, in a Loc≠+2 tamper-

[1] The performance metric we use is the standard Precision/Recall/F measures, as computed by the conlleval program: http://www.cnts.ua.ac.be/conll2000/chunking/conlleval.txt

[2] ftp://ftp.cis.upenn.edu/pub/chunker

[3] http://chasen.org/∼taku/software/yamcha/

ing, features of the form $w_{+2} = X$ are removed).

Loc=i – all the lexical features with an index other than i are removed from the model.

3.6 Results and Discussion

Highlights of the results are presented in Tables (1-3). The numbers reported are F measures.

TopN	HEB$_{Gold}$	HEB$_{Err}$	ENG
ALL	93.58	92.48	93.79
N=0	78.32	76.27	90.10
N=10	90.21	88.68	90.24
N=50	91.78	90.85	91.22
N=100	92.25	91.62	91.72
N=500	93.60	92.23	93.12
N=1000	93.56	92.41	93.30

Table 1: Results of TopN Tampering.

The results of the TopN tamperings show that for both languages, most of the lexical features are irrelevant for the classification – the numbers achieved by using all the lexical features (about 30,000 in Hebrew and 75,000 in English) are very close to those obtained using only a few lexical features. This finding is very encouraging, and suggests that SVM based chunking is robust to corpus variations.

Another conclusion is that lexical features help balance the fact that PoS tags can be noisy: we know both HEB_{Err} and ENG include PoS tagging errors (about 8% in Hebrew and 4% in English). While in the case of "perfect" PoS tagging (HEB_{Gold}), a very small amount of lexical features is sufficient to reach the best F-result (500 out of 30,264), in the presence of PoS errors, more than the top 1000 lexical features are needed to reach the result obtained with all lexical features.

More striking is the fact that in Hebrew, the top 10 lexical features are responsible for an improvement of 12.4 in F-score. The words covered by these 10 features are the following: `Start of Sentence marker` and `comma`, `quote`, '`of/`של', '`and/`ו', '`the/`ה' and '`in/`ב'.

This finding suggests that the Hebrew PoS tagset might not be informative enough for the chunking task, especially where punctuation [4] and prepositions are concerned. The results in Table 2 give further support for this claim.

[4]Unlike the WSJ PoS tagset in which most punctuations get unique tags, our tagset treat punctuation marks as one group.

NoPOS	HEB$_G$	HEB$_E$	NoPOS	HEB$_G$	HEB$_E$
Prep	**85.25**	**84.40**	Pronoun	92.97	92.14
Punct	**88.90**	**87.66**	Conjunction	92.31	91.67
Adverb	92.02	90.72	Determiner	92.55	91.39

Table 2: Results of Hebrew NoPOS Tampering. Other scores are $\geq$ 93.3(HEB$_G$), $\geq$ 92.2(HEB$_E$).

When removing lexical features of a specific PoS, the most dramatic loss of F-score is reached for Prepositions and Punctuation marks, followed by Adverbs, and Conjunctions. Strikingly, lexical information for most open-class PoS (including Proper Names and Nouns) has very little impact on Hebrew chunking performance.

From this observation, one could conclude that enriching a model based only on PoS with lexical features for only a few closed-class PoS (prepositions and punctuation) could provide appropriate results even with a simpler learning method, one that cannot deal with a large number of features. We tested this hypothesis by training the Error-Driven Pruning (EDP) method of (Cardie and Pierce, 1998) with an extended set of features. EDP with PoS features only produced an F-result of 76.3 on HEB_{Gold}. By adding lexical features only for prepositions {של כ ה ב מ}, one conjunction {ו} and punctuation, the F-score on HEB_{Gold} indeed jumps to 85.4. However, when applied on HEB_{Err}, EDP falls down again to 59.4. This striking disparity, by comparison, lets us appreciate the resilience of the SVM model to PoS tagging errors, and its generalization capability even with a reduced number of lexical features.

Another implication of this data is that commas and quotation marks play a major role in determining NP boundaries in Hebrew. In Goldberg *et al.* (2006), we noted the Hebrew Treebank is not consistent in its treatment of punctuation, and thus we evaluated the chunker only after performing normalization of chunk boundaries for punctuations. We now hypothesize that, since commas and quotation marks play such an important role in the classification, performing such normalization *before* the training stage might be beneficial. Indeed results on the normalized corpus show improvement of about 1.0 in F score on both HEB_{Err} and HEB_{Gold}. A 10-fold cross validation experiment on punctuation normalized HEB_{Err} resulted in an F-Score of 92.2, improving the results reported by (Goldberg et al.,

2006) on the same setting (91.4).

Loc=I	HEB$_E$	ENG	Loc≠I	HEB$_E$	ENG
-2	78.26	89.79	-2	91.62	93.87
-1	76.96	90.90	-1	91.86	93.03
0	90.33	92.37	0	79.44	91.16
1	76.90	90.47	1	92.33	93.30
2	76.55	90.06	2	92.18	93.65

Table 3: Results of Loc Tamperings.

We now turn to analyzing the importance of context positions (Table 3). For both languages, the most important lexical feature (by far) is at position 0, that is, the word currently being classified. For English, it is followed by positions 1 and -1, and then positions 2 and -2. For Hebrew, back context seems to have more effect than front context. In Hebrew, all the positions positively contribute to the decision, while in English removing $w_{2/-2}$ slightly improves the results (note also that including only feature $w_{2/-2}$ performs worse than with no lexical information in English).

3.7 The Real Role of Lexical Features

Model tampering (i.e., removing features after the learning stage) is not the same as learning without these features. This claim is verified empirically: training on the English corpus without the lexical features at position –2 yields worse results than with them (93.73 vs. 93.79) – while removing the w_{-2} features via tampering on a model trained with w_{-2} yields better results (93.87). Similarly, for all corpora, training using only the top 1,000 features (as defined in the Top1000 tampering) results in loss of about 2 in F-Score (ENG 92.02, HEB_{Err} 90.30, HEB_{Gold} 91.67), while tampering Top1000 yields a result very close to the best obtained (93.56, 92.41 or 93.3F).

This observation leads us to an interesting conclusion about the real role of lexical features in SVM based chunking: **rare events (features) are used to memorize hard examples**. Intuitively, by giving a heavy weight to rare events, the classifier learns specific rules such as "*if the word at position -2 is X and the PoS at position 2 is Y, then the current word is* **I***nside a noun-phrase*". Most of these rules are accidental – there is no real relation between the particular word-pos combination and the class of the current word, it just happens to be this way in the training samples. Marking the rare occurrences helps the learner achieve better generalization on the other, more common cases, which are similar to the outlier on most features, except the "irrelevant ones". As the events are rare, such rules usually have no effect on chunking accuracy: they simply never occur in the test data. This observation refines the common conception that SVM chunking does not suffer from irrelevant features: in chunking, SVM indeed generalizes well for the common cases but also over-fits the model on outliers.

Model tampering helps us design a model in two ways: (1) it is a way to "open the black box" obtained when training an SVM and to analyze the respective importance of features. In our case, this analysis allowed us to identify the importance of punctuation and prepositions and improve the model by defining more focused features (improving overall result by ∼1.0 F-point). (2) The analysis also led us to the conclusion that "feature selection" is complex in the case of SVM – irrelevant features help prevent over-generalization by forcing over-fitting on outliers.

We have also confirmed that the model learned remains robust in the presence of noise in the PoS tags and relies on only few lexical features. This verification is critical in the context of languages with few computational resources, as we expect the size of corpora and the quality of taggers to keep lagging behind that achieved in English.

4 Anchored Learning

We pursue the observation of how SVM deals with outliers by developing the *Anchored Learning* method. The idea behind Anchored Learning is to add a unique feature a_i (an *anchor*) to each training sample (we add as many new features to the model as there are training samples). These new features make our data linearly separable. The SVM learner can then use these anchors (which will never occur on the test data) to memorize the hard cases, decreasing this burden from "real" features.

We present two uses for Anchored Learning. The first is the identification of hard cases and corpus errors, and the second is a preliminary feature selection approach for SVM to improve chunking accuracy.

4.1 Mining for Errors and Hard Cases

Following the intuition that SVM gives more weight to anchor features of hard-to-classify cases, we can

actively look for such cases by training an SVM chunker on anchored data (as the anchored data is guaranteed to be linearly separable, we can set a very high value to the C parameter, preventing any misclassification), and then investigating either the anchors whose weights[5] are above some threshold t or the top N heaviest anchors, and their corresponding corpus locations. **These locations are those that the learner considers hard to classify**. They can be either corpus errors, or genuinely hard cases.

This method is similar to the corpus error detection method presented by Nakagawa and Matsumoto (2002). They constructed an SVM model for PoS tagging, and considered Support Vectors with high α values to be indicative of suspicious corpus locations. These locations can be either outliers, or correctly labeled locations similar to an outlier. They then looked for similar corpus locations with a different label, to point out right-wrong pairs with high precision.

Using anchors improves their method in three aspects: (1) without anchors, similar examples are often indistinguishable to the SVM learner, and in case they have conflicting labels both examples will be given high weights. That is, both the regular case and the hard case will be considered as hard examples. Moreover, similar corpus errors might result in only one support vector that cover all the group of similar errors. Anchors mitigate these effects, resulting in better precision and recall. (2) The more errors there are in the corpus, the less linearly separable it is. Un-anchored learning on erroneous corpus can take unreasonable amount of time. (3) Anchors allow learning while removing some of the important features but still allow the process to converge in reasonable time. This lets us analyze which cases become hard to learn if we don't use certain features, or in other words: what problematic cases are solved by specific features.

The hard cases analysis achieved by anchored learning is different from the usual error analysis carried out on observed classification errors. The traditional methods give us intuitions about where the classifier **fails to generalize**, while the method we present here gives us intuition about what the classifier **considers hard to learn**, based on the training examples alone.

[5]As each anchor appear in only one support vector, we can treat the vector's α value as the anchor weight

The intuition that "hard to learn" examples are suspect corpus errors is not new, and appears also in Abney *et al.* (1999) , who consider the "heaviest" samples in the final distribution of the AdaBoost algorithm to be the hardest to classify and thus likely corpus errors. While AdaBoost models are easy to interpret, this is not the case with SVM. Anchored learning allows us to extract the hard to learn cases from an SVM model. Interestingly, while both AdaBoost and SVM are 'large margin' based classifiers, there is less than 50% overlap in the hard cases for the two methods (in terms of mistakes on the test data, there were 234 mistakes shared by AdaBoost and SVM, 69 errors unique to SVM and 126 errors unique to AdaBoost)[6]. Analyzing the difference in what the two classifiers consider hard is interesting, and we will address it in future work. In the current work, we note that for finding corpus errors the two methods are complementary.

Experiment 1 – Locating Hard Cases

A linear SVM model (M_{full}) was trained on the training subset of the anchored, punctuation-normalized, HEB_{Gold} corpus, with the same features as in the previous experiments, and a C value of 9,999. Corpus locations corresponding to anchors with weights >1 were inspected. There were about 120 such locations out of 4,500 sentences used in the training set. Decreasing the threshold t would result in more cases. We analyzed these locations into 3 categories: corpus errors, cases that challenge the SimpleNP definition, and cases where the chunking decision is genuinely difficult to make in the absence of global syntactic context or world knowledge.

Corpus Errors: The analysis revealed the following corpus errors: we identified 29 hard cases related to conjunction and apposition (is the comma, colon or slash inside an NP or separating two distinct NPs). 14 of these hard cases were indeed mistakes in the corpus. This was anticipated, as we distinguished appositions and conjunctive commas using heuristics, since the Treebank marking of conjunctions is somewhat inconsistent.

In order to build the Chunk NP corpus, the syntactic trees of the Treebank were processed to derive chunks according to the SimpleNP definition. The hard cases analysis identified 18 instances where this

[6]These numbers are for pairwise Linear SVM and AdaBoost classifiers trained on the same features.

transformation results in erroneous chunks. For example, null elements result in improper chunks, such as chunks containing only adverbs or only adjectives.

We also found 3 invalid sentences, 6 inconsistencies in the tagging of interrogatives with respect to chunk boundaries, as well as 34 other specific mistakes. Overall, more than half of the locations identified by the anchors were corpus errors. Looking for cases similar to the errors identified by anchors, we found 99 more locations, 77 of which were errors.

Refining the SimpleNP Definition: The hard cases analysis identified examples that challenge the SimpleNP definition proposed in Goldberg *et al.* (2006). The most notable cases are:

The 'et' marker : '*et*' is a syntactic marker of definite direct objects in Hebrew. It was regarded as a part of SimpleNPs in their definition. In some cases, this forces the resulting SimpleNP to be too inclusive:

[את הממשלה, הכנסת בית המשפט והתקשורת]

['et' (the government, the parliament and the media)]

Because in the Treebank the conjunction depends on '*et*' as a single constituent, it is fully embedded in the chunk. Such a conjunction should not be considered simple.

The של *preposition ('of')* marks generalized possession and was considered unambiguous and included in SimpleNPs. We found cases where 'של' causes PP attachment ambiguity:

[נשיא בית הדין] ל [משמעת] של [המשטרה]

[president-cons house-cons the-law] for [discipline] of [the police] / The Police Disciplinary Court President

Because 2 prepositions are involved in this NP, 'של' (*of*) and 'ל' (*for*), the 'של' part cannot be attached unambiguously to its head ('*court*'). It is unclear whether the 'ל' preposition should be given special treatment to allow it to enter simple NPs in certain contexts, or whether the inconsistent handling of the 'של' that results from the 'ל' inter-position is preferable.

Complex determiners and quantifiers: In many cases, complex determiners in Hebrew are multi-word expressions that include nouns. The inclusion of such determiners inside the SimpleNPs is not consistent.

Genuinely hard cases were also identified. These include prepositions, conjunctions and multi-word idioms (most of them are adjectives and prepositions which are made up of nouns and determiners, *e.g.*, as the word *unanimously* is expressed in Hebrew as the multi-word expression 'one mouth'). Also, some *adverbials* and *adjectives* are impossible to distinguish using only local context.

The anchors analysis helped us improve the chunking method on two accounts: (1) it identified corpus errors with high precision; (2) it made us focus on hard cases that challenge the linguistic definition of chunks we have adopted. Following these findings, we intend to refine the Hebrew SimpleNP definition, and create a new version of the Hebrew chunking corpus.

Experiment 2 – determining the role of contextual lexical features

The intent of this experiment is to understand the role of the contextual lexical features (w_i, $i \neq 0$). This is done by training 2 additional anchored linear SVM models, $M_{no-cont}$ and M_{near}. These are the same as M_{full} except for the lexical features used during training. $M_{no-cont}$ uses only w_0, while M_{near} uses w_0,w_{-1},w_{+1}.

Anchors are again used to locate the hard examples for each classifier, and the differences are examined. The examples that are hard for M_{near} but not for M_{full} are those solved by w_{-2},w_{+2}. Similarly, the examples that are hard for $M_{no-cont}$ but not for M_{near} are those solved by w_{-1},w_{+1}. Table 4 indicates the number of hard cases identified by the anchor method for each model. One way to interpret these figures, is that the introduction of features $w_{-1,+1}$ solves 5 times more hard cases than $w_{-2,+2}$.

Model	**Number of hard cases ($t = 1$)**	**Hard cases for classifier B-I**
M_{full}	120	2
M_{near}	320 (+ 200)	12
$M_{no-cont}$	1360 (+ 1040)	164

Table 4: Number of hard cases per model type.

Qualitative analysis of the hard cases solved by the contextual lexical features shows that they contribute mostly to the identification of chunk boundaries in cases of conjunction, apposition, attachment of adverbs and adjectives, and some multi-word expressions.

The number of hard cases specific to the B-I classifier indicates how the features contribute to the decision of splitting or continuing back-to-back NPs. Back-to-back NPs amount to 6% of the NPs in HEB_{Gold} and 8% of the NPs in ENG. However,

while in English most of these cases are easily resolved, Hebrew phenomena such as null-equatives and free word order make them harder. To quantify the difference: 79% of the first words of the second NP in English belong to one of the closed classes POS, DT, WDT, PRP, WP – categories which mostly cannot appear in the middle of base NPs. In contrast, in Hebrew, 59% are Nouns, Numbers or Proper Names. Moreover, in English the ratio of unique first words to number of adjacent NPs is 0.068, while in Hebrew it is 0.47. That is, in Hebrew, almost every second such NP starts with a different word.

These figures explain why surrounding lexical information is needed by the learner in order to classify such cases. They also suggest that this learning is mostly superficial, that is, the learner just memorizes some examples, but these will not generalize well on test data. Indeed, the most common class of errors reported in Goldberg *et al.* , 2006 are of the split/merge type. These are followed by conjunction related errors, which suffer from the same problem. Morphological features of *smixut* and agreement can help to some extent, but this is still a limited solution. It seems that deciding the [NP][NP] case is beyond the capabilities of chunking with local context features alone, and more global features should be sought.

4.2 Facilitating Better Learning

This section presents preliminary results using Anchored Learning for better NP chunking. We present a setting (English Base NP chunking) in which selected features coupled together with anchored learning show an improvement over previous results.

Section 3.6 hinted that SVM based chunking might be hurt by using too many lexical features. Specifically, the features w_{-2},w_{+2} were shown to cause the chunker to overfit in English chunking. Learning without these features, however, yields lower results. This can be overcome by introducing anchors as a substitute. Anchors play the same role as rare features when learning, while lowering the chance of misleading the classifier on test data.

The results of the experiment using 5-fold cross validation on ENG indicate that the F-score improves on average from 93.95 to 94.10 when using anchors instead of $w_{\pm 2}$ (+0.15), while just ignoring the $w_{\pm 2}$ features drops the F-score by 0.10. The improvement is minor but consistent. Its implication is that anchors can substitute for "irrelevant" lexical features for better learning results. In future work, we will experiment with better informed sets of lexical features mixed with anchors.

5 Conclusion

We have introduced two novel methods to understand the inner structure of SVM-learned models. We have applied these techniques to Hebrew NP chunking, and demonstrated that the learned model is robust in the presence of noise in the PoS tags, and relies on only a few lexical features. We have identified corpus errors, better understood the nature of the task in Hebrew – and compared it quantitatively to the task in English.

The methods provide general insight in the way SVM classification works for chunking.

References

S. Abney, R. Schapire, and Y. Singer. 1999. Boosting applied to tagging and PP attachment. *EMNLP-1999*.

M. Adler and M. Elhadad. 2006. An unsupervised morpheme-based hmm for hebrew morphological disambiguation. In *COLING/ACL2006*.

C. Cardie and D. Pierce. 1998. Error-driven pruning of treebank grammars for base noun phrase identification. In *ACL-1998*.

Y. Goldberg, M. Adler, and M. Elhadad. 2006. Noun phrase chunking in hebrew: Influence of lexical and morphological features. In *COLING/ACL2006*.

T. Kudo and Y. Matsumoto. 2000. Use of support vector learning for chunk identification. In *CoNLL-2000*.

T. Kudo and Y. Matsumoto. 2003. Fast methods for kernel-based text analysis. In *ACL-2003*.

M. Marcus and L. Ramshaw. 1995. Text Chunking Using Transformation-Based Learning. In *Proc. of the 3rd ACL Workshop on Very Large Corpora*.

T. Nakagawa and Y. Matsumoto. 2002. Detecting errors in corpora using support vector machines. In *COLING-2002*.

Erik F. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the conll-2000 shared task: chunking. In *CoNLL-2000*.

K. Sima'an, A. Itai, Y. Winter, A. Altman, and N. Nativ. 2001. Building a tree-bank of modern hebrew text. *Traitement Automatique des Langues*, 42(2).

V. Vapnik. 1995. *The nature of statistical learning theory*. Springer-Verlag New York, Inc.

Fully Unsupervised Discovery of Concept-Specific Relationships by Web Mining

Dmitry Davidov
ICNC
The Hebrew University
Jerusalem 91904, Israel
dmitry@alice.nc.huji.ac.il

Ari Rappoport
Institute of Computer Science
The Hebrew University
Jerusalem 91904, Israel
www.cs.huji.ac.il/∼arir

Moshe Koppel
Dept. of Computer Science
Bar-Ilan University
Ramat-Gan 52900, Israel
koppel@cs.biu.ac.il

Abstract

We present a web mining method for discovering and enhancing relationships in which a specified concept (word class) participates. We discover a whole range of relationships focused on the given concept, rather than generic known relationships as in most previous work. Our method is based on clustering patterns that contain concept words and other words related to them. We evaluate the method on three different rich concepts and find that in each case the method generates a broad variety of relationships with good precision.

1 Introduction

The huge amount of information available on the web has led to a flurry of research on methods for automatic creation of structured information from large unstructured text corpora. The challenge is to create as much information as possible while providing as little input as possible.

A lot of this research is based on the initial insight (Hearst, 1992) that certain lexical patterns ('X is a country') can be exploited to automatically generate hyponyms of a specified word. Subsequent work (to be discussed in detail below) extended this initial idea along two dimensions.

One objective was to require as small a user-provided initial seed as possible. Thus, it was observed that given one or more such lexical patterns, a corpus could be used to generate examples of hyponyms that could then, in turn, be exploited to generate more lexical patterns. The larger and more reliable sets of patterns thus generated resulted in larger and more precise sets of hyponyms and vice versa. The initial step of the resulting alternating bootstrap process – the user-provided input – could just as well consist of examples of hyponyms as of lexical patterns.

A second objective was to extend the information that could be learned from the process beyond hyponyms of a given word. Thus, the approach was extended to finding lexical patterns that could produce synonyms and other standard lexical relations. These relations comprise all those words that stand in some known binary relation with a specified word.

In this paper, we introduce a novel extension of this problem: given a particular concept (initially represented by two seed words), discover relations in which it participates, without specifying their types in advance. We will generate a concept class and a variety of natural binary relations involving that class.

An advantage of our method is that it is particularly suitable for web mining, even given the restrictions on query amounts that exist in some of today's leading search engines.

The outline of the paper is as follows. In the next section we will define more precisely the problem we intend to solve. In section 3, we will consider related work. In section 4 we will provide an overview of our solution and in section 5 we will consider the details of the method. In section 6 we will illustrate and evaluate the results obtained by our method. Finally, in section 7 we will offer some conclusions and considerations for further work.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 232–239,
Prague, Czech Republic, June 2007.

2 Problem Definition

In several studies (e.g., Widdows and Dorow, 2002; Pantel et al, 2004; Davidov and Rappoport, 2006) it has been shown that relatively unsupervised and language-independent methods could be used to generate many thousands of sets of words whose semantics is similar in some sense. Although examination of any such set invariably makes it clear why these words have been grouped together into a single concept, it is important to emphasize that the method itself provides no explicit concept definition; in some sense, the implied class is in the eye of the beholder. Nevertheless, both human judgment and comparison with standard lists indicate that the generated sets correspond to concepts with high precision.

We wish now to build on that result in the following way. Given a large corpus (such as the web) and two or more examples of some concept X, automatically generate examples of one or more relations $R \subset X \times Y$, where Y is some concept and R is some binary relationship between elements of X and elements of Y.

We can think of the relations we wish to generate as bipartite graphs. Unlike most earlier work, the bipartite graphs we wish to generate might be one-to-one (for example, countries and their capitals), many-to-one (for example, countries and the regions they are in) or many-to-many (for example, countries and the products they manufacture). For a given class X, we would like to generate not one but possibly many different such relations.

The only input we require, aside from a corpus, is a small set of examples of some class. However, since such sets can be generated in entirely unsupervised fashion, our challenge is effectively to generate relations directly from a corpus given no additional information of any kind. The key point is that we do not in any manner specify in advance what types of relations we wish to find.

3 Related Work

As far as we know, no previous work has directly addressed the discovery of generic binary relations in an unrestricted domain without (at least implicitly) pre-specifying relationship types. Most related work deals with discovery of hypernymy (Hearst, 1992; Pantel et al, 2004), synonymy (Roark and Charniak, 1998; Widdows and Dorow, 2002; Davidov and Rappoport, 2006) and meronymy (Berland and Charniak, 1999).

In addition to these basic types, several studies deal with the discovery and labeling of more specific relation sub-types, including inter-verb relations (Chklovski and Pantel, 2004) and noun-compound relationships (Moldovan et al, 2004).

Studying relationships between tagged named entities, (Hasegawa et al, 2004; Hassan et al, 2006) proposed unsupervised clustering methods that assign given (or semi-automatically extracted) sets of pairs into several clusters, where each cluster corresponds to one of a known relationship type. These studies, however, focused on the classification of pairs that were either given or extracted using some supervision, rather than on discovery and definition of which relationships are actually in the corpus.

Several papers report on methods for using the web to discover instances of binary relations. However, each of these assumes that the relations themselves are known in advance (implicitly or explicitly) so that the method can be provided with seed patterns (Agichtein and Gravano, 2000; Pantel et al, 2004), pattern-based rules (Etzioni et al, 2004), relation keywords (Sekine, 2006), or word pairs exemplifying relation instances (Pasca et al, 2006; Alfonseca et al, 2006; Rosenfeld and Feldman, 2006).

In some recent work (Strube and Ponzetto, 2006), it has been shown that related pairs can be generated without pre-specifying the nature of the relation sought. However, this work does not focus on differentiating among different relations, so that the generated relations might conflate a number of distinct ones.

It should be noted that some of these papers utilize language and domain-dependent preprocessing including syntactic parsing (Suchanek et al, 2006) and named entity tagging (Hasegawa et al, 2004), while others take advantage of handcrafted databases such as WordNet (Moldovan et al, 2004; Costello et al, 2006) and Wikipedia (Strube and Ponzetto, 2006).

Finally, (Turney, 2006) provided a pattern distance measure which allows a fully unsupervised measurement of relational similarity between two pairs of words; however, relationship types were not discovered explicitly.

4 Outline of the Method

We will use two concept words contained in a concept class C to generate a collection of distinct relations in which C participates. In this section we offer a brief overview of our method.

Step 1: Use a seed consisting of two (or more) example words to automatically obtain other examples that belong to the same class. Call these *concept words*. (For instance, if our example words were *France* and *Angola*, we would generate more country names.)

Step 2: For each concept word, collect instances of contexts in which the word appears together with one other content word. Call this other word a *target word* for that concept word. (For example, for *France* we might find 'Paris is the capital of France'. *Paris* would be a target word for *France*.)

Step 3: For each concept word, group the contexts in which it appears according to the target word that appears in the context. (Thus 'X is the capital of Y' would likely be grouped with 'Y's capital is X'.)

Step 4: Identify similar context groups that appear across many different concept words. Merge these into a single concept-word-independent cluster. (The group including the two contexts above would appear, with some variation, for other countries as well, and all these would be merged into a single cluster representing the relation *capital-of(X,Y)*.)

Step 5: For each cluster, output the relation consisting of all <concept word, target word> pairs that appear together in a context included in the cluster. (The cluster considered above would result in a set of pairs consisting of a country and its capital. Other clusters generated by the same seed might include countries and their languages, countries and the regions in which they are located, and so forth.)

5 Details of the Method

In this section we consider the details of each of the above-enumerated steps. It should be noted that each step can be performed using standard web searches; no special pre-processed corpus is required.

5.1 Generalizing the seed

The first step is to take the seed, which might consist of as few as two concept words, and generate many (ideally, all, when the concept is a closed set of words) members of the class to which they belong. We do this as follows, essentially implementing a simplified version of the method of Davidov and Rappoport (2006). For any pair of seed words S_i and S_j, search the corpus for word patterns of the form S_iHS_j, where H is a high-frequency word in the corpus (we used the 100 most frequent words in the corpus). Of these, we keep all those patterns, which we call *symmetric patterns*, for which S_jHS_i is also found in the corpus. Repeat this process to find symmetric patterns with any of the structures $HSHS$, $SHSH$ or $SHHS$. It was shown in (Davidov and Rappoport, 2006) that pairs of words that often appear together in such symmetric patterns tend to belong to the same class (that is, they share some notable aspect of their semantics). Other words in the class can thus be generated by searching a sub-corpus of documents including at least two concept words for those words X that appear in a sufficient number of instances of both the patterns S_iHX and XHS_i, where S_i is a word in the class. The same can be done for the other three pattern structures. The process can be bootstrapped as more words are added to the class.

Note that our method differs from that of Davidov and Rappoport (2006) in that here we provide an initial seed pair, representing our target concept, while there the goal is grouping of as many words as possible into concept classes. The focus of our paper is on relations involving a specific concept.

5.2 Collecting contexts

For each concept word S, we search the corpus for distinct contexts in which S appears. (For our purposes, a context is a window with exactly five words or punctuation marks before or after the concept word; we choose 10,000 of these, if available.) We call the aggregate text found in all these context windows the S-corpus.

From among these contexts, we choose all patterns of the form $H_1SH_2XH_3$ or $H_1XH_2SH_3$, where:

- X is a word that appears with frequency below f_1 in the S-corpus and that has sufficiently high pointwise mutual information with S. We use these two criteria to ensure that X is a content word and that it is related to S. The lower the threshold f_1, the less noise we allow in, though possibly at the expense of recall. We used $f_1 = 1,000$ occurrences per million words.

- H_2 is a string of words each of which occurs with frequency above f_2 in the S-corpus. We want H_2 to consist mainly of words common in the context of S in order to restrict patterns to those that are somewhat generic. Thus, in the context of countries we would like to retain words like *capital* while eliminating more specific words that are unlikely to express generic patterns. We used $f_2 = 100$ occurrences per million words (there is room here for automatic optimization, of course).

- H_1 and H_3 are either punctuation or words that occur with frequency above f_3 in the S-corpus. This is mainly to ensure that X and S aren't fragments of multi-word expressions. We used $f_3 = 100$ occurrences per million words.

- We call these patterns, *S-patterns* and we call X the *target* of the S-pattern. The idea is that S and X very likely stand in some fixed relation to each other where that relation is captured by the S-pattern.

5.3 Grouping S-patterns

If S is in fact related to X in some way, there might be a number of S-patterns that capture this relationship. For each X, we group all the S-patterns that have X as a target. (Note that two S-patterns with two different targets might be otherwise identical, so that essentially the same pattern might appear in two different groups.) We now merge groups with large (more than 2/3) overlap. We call the resulting groups, *S-groups.*

5.4 Identifying pattern clusters

If the S-patterns in a given S-group actually capture some relationship between S and the target, then one would expect that similar groups would appear for a multiplicity of concept words S. Suppose that we have S-groups for three different concept words S such that the pairwise overlap among the three groups is more than 2/3 (where for this purpose two patterns are deemed identical if they differ only at S and X). Then the set of patterns that appear in two or three of these S-groups is called a *cluster core.* We now group all patterns in other S-groups that have an overlap of more than 2/3 with the cluster core into a candidate pattern pool P. The set of all patterns in P that appear in at least two S-groups (among those that formed P) *pattern cluster.* A pattern cluster that has patterns instantiated by at least half of the concept words is said to represent a relation.

5.5 Refining relations

A relation consists of pairs (S, X) where S is a concept word and X is the target of some S-pattern in a given pattern cluster. Note that for a given S, there might be one or many values of X satisfying the relation. As a final refinement, for each given S, we rank all such X according to pointwise mutual information with S and retain only the highest 2/3. If most values of S have only a single corresponding X satisfying the relation and the rest have none, we try to automatically fill in the missing values by searching the corpus for relevant S-patterns for the missing values of S. (In our case the corpus is the web, so we perform additional clarifying queries.)

Finally, we delete all relations in which all concept words are related to most target words and all relations in which the concept words and the target words are identical. Such relations can certainly be of interest (see Section 7), but are not our focus in this paper.

5.6 Notes on required Web resources

In our implementation we use the Google search engine. Google restricts individual users to 1,000 queries per day and 1,000 pages per query. In each stage we conducted queries iteratively, each time downloading all 1,000 documents for the query.

In the first stage our goal was to discover symmetric relationships from the web and consequently discover additional concept words. For queries in this stage of our algorithm we invoked two requirements.

First, the query should contain at least two concept words. This proved very effective in reduc-

ing ambiguity. Thus of 1,000 documents for the query *bass*, 760 deal with music, while if we add to the query a second word from the intended concept (e.g., *barracuda*), then none of the 1,000 documents deal with music and the vast majority deal with fish, as intended.

Second, we avoid doing overlapping queries. To do this we used Google's ability to exclude from search results those pages containing a given term (in our case, one of the concept words).

We performed up to 300 different queries for individual concepts in the first stage of our algorithm.

In the second stage, we used web queries to assemble S-corpora. On average, about 1/3 of the concept words initially lacked sufficient data and we performed up to twenty additional queries for each rare concept word to fill its corpus.

In the last stage, when clusters are constructed, we used web queries for filling missing pairs of one-to-one or several-to-several relationships. The total number of filling queries for a specific concept was below 1,000, and we needed only the first results of these queries. Empirically, it took between 0.5 to 6 day limits (i.e., 500–6,000 queries) to extract relationships for a concept, depending on its size (the number of documents used for each query was at most 100). Obviously this strategy can be improved by focused crawling from primary Google hits, which can drastically reduce the required number of queries.

6 Evaluation

In this section we wish to consider the variety of relations that can be generated by our method from a given seed and to measure the quality of these relations in terms of their precision and recall.

With regard to precision, two claims are being made. One is that the generated relations correspond to identifiable relations. The other claim is that to the extent that a generated relation can be reasonably identified, the generated pairs do indeed belong to the identified relation. (There is a small degree of circularity in this characterization but this is probably the best we can hope for.)

As a practical matter, it is extremely difficult to measure precision and recall for relations that have not been pre-determined in any way. For each generated relation, authoritative resources must be marshaled as a gold standard. For purposes of evaluation, we ran our algorithm on three representative domains – countries, fish species and star constellations – and tracked down gold standard resources (encyclopedias, academic texts, informative websites, etc) for the bulk of the relations generated in each domain.

This choice of domains allowed us to explore different aspects of algorithmic behavior. Country and constellation domains are both well defined and closed domains. However they are substantially different.

Country names is a relatively large domain which has very low lexical ambiguity, and a large number of potentially useful relations. The main challenge in this domain was to capture it well.

Constellation names, in contrast, are a relatively small but highly ambiguous domain. They are used in proper names, mythology, names of entertainment facilities etc. Our evaluation examined how well the algorithm can deal with such ambiguity.

The fish domain contains a very high number of members. Unlike countries, it is a semi-open non-homogenous domain with a very large number of subclasses and groups. Also, unlike countries, it does not contain many proper nouns, which are empirically generally easier to identify in patterns. So the main challenge in this domain is to extract unblurred relationships and not to diverge from the domain during the concept acquisition phase.

We do not show here all-to-all relationships such as fish parts (common to all or almost all fish), because we focus on relationships that separate between members of the concept class, which are harder to acquire and evaluate.

6.1 Countries

Our seed consisted of two country names. The intended result for the first stage of the algorithm was a list of countries. There are 193 countries in the world (www.countrywatch.com) some of which have multiple names so that the total number of commonly used country names is 243. Of these, 223 names (comprising 180 countries) are character strings with no white space. Since we consider only single word names, these 223 are the names we hope to capture in this stage.

Using the seed words *France* and *Angola*, we obtained 202 country names (comprising 167 distinct countries) as well as 32 other names (consisting mostly of names of other geopolitical entities). Using the list of 223 single word countries as our gold standard, this gives precision of 0.90 and recall of 0.86. (Ten other seed pairs gave results ranging in precision: 0.86-0.93 and recall: 0.79-0.90.)

The second part of the algorithm generated a set of 31 binary relations. Of these, 25 were clearly identifiable relations many of which are shown in Table 1. Note that for three of these there are standard exhaustive lists against which we could measure both precision and recall; for the others shown, sources were available for measuring precision but no exhaustive list was available from which to measure recall, so we measured coverage (the number of countries for which at least one target concept is found as related).

Another eleven meaningful relations were generated for which we did not compute precision numbers. These include *celebrity-from, animal-of, lake-in, borders-on* and *enemy-of.* (The set of relations generated by other seed pairs differed only slightly from those shown here for *France* and *Angola*.)

6.2 Fish species

In our second experiment, our seed consisted of two fish species, *barracuda* and *bluefish.* There are 770 species listed in WordNet of which 447 names are character strings with no white space. The first stage of the algorithm returned 305 of the species listed in Wordnet, another 37 species not listed in Wordnet, as well as 48 other names (consisting mostly of other sea creatures). The second part of the algorithm generated a set of 15 binary relations all of which are meaningful. Those for which we could find some gold standard are listed in Table 2.

Other relations generated include *served-with, bait-for, food-type, spot-type,* and *gill-type.*

6.3 Constellations

Our seed consisted of two constellation names, *Orion* and *Cassiopeia.* There are 88 standard constellations (www.astro.wisc.edu) some of which have multiple names so that the total number of commonly used constellations is 98. Of these, 87 names (77 constellations) are strings with no white space.

Relationship *Sample pattern* (Sample pair)	Prec.	Rec/Cov
capital-of *in (x), capital of (y),* (Luanda, Angola)	0.92	R=0.79
language-spoken-in *to (x) or other (y) speaking* (Spain, Spanish)	0.92	R=0.60
in-region *throughout (x), from (y) to* (America, Canada)	0.73	R=0.71
city-in *west (x) – forecast for (y).* (England, London)	0.82	C=0.95
river-in *central (x), on the (y) river* (China, Haine)	0.92	C=0.68
mountain-range-in *the (x) mountains in (y) ,* (Chella, Angola)	0.77	C=0.69
sub-region-of *the (y) region of (x),* (Veneto, Italy)	0.81	C=0.81
industry-of *the (x) industry in (y) ,* (Oil, Russia)	0.70	C=0.90
island-in *, (x) island , (y) ,* (Bathurst, Canada)	0.98	C=0.55
president-of *president (x) of (y) has* (Bush, USA)	0.86	C=0.51
political-position-in *former (x) of (y) face* (President, Ecuador)	0.81	C=0.75
political-party-of *the (x) party of (y) ,* (Labour, England)	0.91	C=0.53
festival-of *the (x) festival, (y) ,* (Tanabata, Japan)	0.90	C=0.78
religious-denomination-of *the (x) church in (y) ,* (Christian, Rome)	0.80	C=0.62

Table 1: Results on seed { *France, Angola* }.

Relationship *Sample pattern* (Sample pair)	Prec.	Cov
region-found-in *best (x) fishing in (y) .* (Walleye, Canada)	0.83	0.80
sea-found-in *of (x) catches in the (y) sea* (Shark, Adriatic)	0.82	0.64
lake-found-in *lake (y) is famous for (x) ,* (Marion, Catfish)	0.79	0.51
habitat-of *, (x) and other (y) fish* (Menhaden, Saltwater)	0.78	0.92
also-called *. (y) , also called (x) ,* (Lemonfish, Ling)	0.91	0.58
eats *the (x) eats the (y) and* (Perch, Minnow)	0.90	0.85
color-of *the (x) was (y) color* (Shark, Gray)	0.95	0.85
used-for-food *catch (x) – best for (y) or* (Bluefish, Sashimi)	0.80	0.53
in-family *the (x) family , includes (y) ,* (Salmonid, Trout)	0.95	0.60

Table 2: Results on seed { *barracud, bluefish* }.

The first stage of the algorithm returned 81 constellation names (77 distinct constellations) as well as 38 other names (consisting mostly of names of individual stars). Using the list of 87 single word constellation names as our gold standard, this gives precision of 0.68 and recall of 0.93.

The second part of the algorithm generated a set of ten binary relations. Of these, one concerned travel and entertainment (constellations are quite popular as names of hotels and lounges) and another three were not interesting. Apparently, the requirement that half the constellations appear in a relation limited the number of viable relations since many constellations are quite obscure. The six interesting relations are shown in Table 3 along with precision and coverage.

7 Discussion

In this paper we have addressed a novel type of problem: given a specific concept, discover in fully unsupervised fashion, a range of relations in which it participates. This can be extremely useful for studying and researching a particular concept or field of study.

As others have shown as well, two concept words can be sufficient to generate almost the entire class to which the words belong when the class is well-defined. With the method presented in this paper, using no further user-provided information, we can, for a given concept, automatically generate a diverse collection of binary relations on this concept. These relations need not be pre-specified in any way. Results on the three domains we considered indicate that, taken as an aggregate, the relations that are generated for a given domain paint a rather clear picture of the range of information pertinent to that domain.

Moreover, all this was done using standard search engine methods on the web. No language-dependent tools were used (not even stemming); in fact, we reproduced many of our results using Google in Russian.

The method depends on a number of numerical parameters that control the subtle tradeoff between quantity and quality of generated relations. There is certainly much room for tuning of these parameters.

The concept and target words used in this paper are single words. Extending this to multiple-word expressions would substantially contribute to the applicability of our results.

In this research we effectively disregard many relationships of an all-to-all nature. However, such relationships can often be very useful for ontology construction, since in many cases they introduce strong connections between two different concepts. Thus, for fish we discovered that one of the all-to-all relationships captures a precise set of fish body parts, and another captures swimming verbs. Such relations introduce strong and distinct connections between the concept of fish and the concepts of fish-body-parts and swimming. Such connections may be extremely useful for ontology construction.

Relationship *Sample pattern* (Sample pair)	Prec.	Cov
nearby-constellation *constellation (x), near (y),* (Auriga, Taurus)	0.87	0.70
star-in *star (x) in (y) is* (Antares , Scorpius)	0.82	0.76
shape-of *, (x) is depicted as (y).* (Lacerta, Lizard)	0.90	0.55
abbreviated-as *. (x) abbr (y),* (Hidra, Hya)	0.93	0.90
cluster-types-in *famous (x) cluster in (y),* (Praesepe, Cancer)	0.92	1.00
location *, (x) is a (y) constellation* (Draco, Circumpolar)	0.82	0.70

Table 3: Results on seed { *Orion, Cassiopeia* }.

References

Agichtein, E., Gravano, L., 2000. Snowball: Extracting relations from large plain-text collections. Proceedings of the 5th ACM International Conference on Digital Libraries.

Alfonseca, E., Ruiz-Casado, M., Okumura, M., Castells, P., 2006. Towards large-scale non-taxonomic relation extraction: estimating the precision of rote extractors. Workshop on Ontology Learning and Population at COLING-ACL '06.

Berland, M., Charniak, E., 1999. Finding parts in very large corpora. ACL '99.

Chklovski T., Pantel P., 2004. VerbOcean: mining the web for fine-grained semantic verb relations. EMNLP '04.

Costello, F., Veale, T., Dunne, S., 2006. Using WordNet to automatically deduce relations between words in noun-noun compounds, COLING-ACL '06.

Davidov, D., Rappoport, A., 2006. Efficient unsupervised discovery of word categories using symmetric patterns and high frequency words. COLING-ACL '06.

Etzioni, O., Cafarella, M., Downey, D., Popescu, A., Shaked, T., Soderland, S., Weld, D., Yates, A., 2004. Methods for domain-independent information extraction from the web: an experimental comparison. AAAI '04.

Hasegawa, T., Sekine, S., Grishman, R., 2004. Discovering relations among named entities from large corpora. ACL '04.

Hassan, H., Hassan, A., Emam, O., 2006. unsupervised information extraction approach using graph mutual reinforcement. EMNLP '06.

Hearst, M., 1992. Automatic acquisition of hyponyms from large text corpora. COLING '92.

Moldovan, D., Badulescu, A., Tatu, M., Antohe, D., Girju, R., 2004. Models for the semantic classification of noun phrases. Workshop on Comput. Lexical Semantics at HLT-NAACL '04.

Pantel, P., Ravichandran, D., Hovy, E., 2004. Towards terascale knowledge acquisition. COLING '04.

Pasca, M., Lin, D., Bigham, J., Lifchits A., Jain, A., 2006. Names and similarities on the web: fact extraction in the fast lane. COLING-ACL '06.

Roark, B., Charniak, E., 1998. Noun-phrase co-occurrence statistics for semi-automatic semantic lexicon construction. ACL '98.

Rosenfeld B., Feldman, R.: URES : an unsupervised web relation extraction system. Proceedings, ACL '06 Poster Sessions.

Sekine, S., 2006 On-demand information extraction. COLING-ACL '06.

Strube, M., Ponzetto, S., 2006. WikiRelate! computing semantic relatedness using Wikipedia. AAAI '06.

Suchanek F. M., G. Ifrim, G. Weikum. 2006. LEILA: learning to extract information by linguistic analysis. Workshop on Ontology Learning and Population at COLING-ACL '06.

Turney, P., 2006. Expressing implicit semantic relations without supervision. COLING-ACL '06.

Widdows, D., Dorow, B., 2002. A graph model for unsupervised Lexical acquisition. COLING '02.

Adding Noun Phrase Structure to the Penn Treebank

David Vadas and **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{dvadas1, james}@it.usyd.edu.au

Abstract

The Penn Treebank does not annotate within base noun phrases (NPs), committing only to flat structures that ignore the complexity of English NPs. This means that tools trained on Treebank data cannot learn the correct internal structure of NPs.

This paper details the process of adding gold-standard bracketing within each noun phrase in the Penn Treebank. We then examine the consistency and reliability of our annotations. Finally, we use this resource to determine NP structure using several statistical approaches, thus demonstrating the utility of the corpus. This adds detail to the Penn Treebank that is necessary for many NLP applications.

1 Introduction

The Penn Treebank (Marcus et al., 1993) is perhaps the most influential resource in Natural Language Processing (NLP). It is used as a standard training and evaluation corpus in many syntactic analysis tasks, ranging from part of speech (POS) tagging and chunking, to full parsing.

Unfortunately, the Penn Treebank does not annotate the internal structure of base noun phrases, instead leaving them flat. This significantly simplified and sped up the manual annotation process.

Therefore, any system trained on Penn Treebank data will be unable to model the syntactic and semantic structure inside base-NPs.

The following NP is an example of the flat structure of base-NPs within the Penn Treebank:

```
(NP (NNP Air) (NNP Force) (NN contract))
```

Air Force is a specific entity and should form a separate constituent underneath the NP, as in our new annotation scheme:

```
(NP
  (NML (NNP Air) (NNP Force))
  (NN contract))
```

We use NML to specify that *Air Force* together is a nominal modifier of *contract*. Adding this annotation better represents the true syntactic and semantic structure, which will improve the performance of downstream NLP systems.

Our main contribution is a gold-standard labelled bracketing for every ambiguous noun phrase in the Penn Treebank. We describe the annotation guidelines and process, including the use of named entity data to improve annotation quality. We check the correctness of the corpus by measuring inter-annotator agreement, by reannotating the first section, and by comparing against the sub-NP structure in DepBank (King et al., 2003).

We also give an analysis of our extended Treebank, quantifying how much structure we have added, and how it is distributed across NPs. Finally, we test the utility of the extended Treebank for training statistical models on two tasks: NP bracketing (Lauer, 1995; Nakov and Hearst, 2005) and full parsing (Collins, 1999).

This new resource will allow any system or annotated corpus developed from the Penn Treebank, to represent noun phrase structure more accurately.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 240–247,
Prague, Czech Republic, June 2007.

2 Motivation

Many approaches to identifying base noun phrases have been explored as part of chunking (Ramshaw and Marcus, 1995), but determining sub-NP structure is rarely addressed. We could use multi-word expressions (MWEs) to identify some structure. For example, knowing *stock market* is a MWE may help bracket *stock market prices* correctly, and Named Entities (NEs) can be used the same way. However, this only resolves NPs dominating MWEs or NEs.

Understanding base-NP structure is important, since otherwise parsers will propose nonsensical noun phrases like *Force contract* by default and pass them onto downstream components. For example, Question Answering (QA) systems need to supply an NP as the answer to a factoid question, often using a parser to identify candidate NPs to return to the user. If the parser never generates the correct sub-NP structure, then the system may return a nonsensical answer even though the correct dominating noun phrase has been found.

Base-NP structure is also important for annotated data derived from the Penn Treebank. For instance, CCGbank (Hockenmaier, 2003) was created by semi-automatically converting the Treebank phrase structure to Combinatory Categorial Grammar (CCG) (Steedman, 2000) derivations. Since CCG derivations are binary branching, they cannot directly represent the flat structure of the Penn Treebank base-NPs.

Without the correct bracketing in the Treebank, strictly right-branching trees were created for all base-NPs. This has an unwelcome effect when conjunctions occur within an NP (Figure 1). An additional grammar rule is needed just to get a parse, but it is still not correct (Hockenmaier, 2003, p. 64). The awkward conversion results in bracketing (a) which should be (b):

(a) *(consumer ((electronics) and (appliances (retailing chain))))*

(b) *((((consumer electronics) and appliances) retailing) chain)*

We have previously experimented with using NEs to improve parsing performance on CCGbank. Due to the mis-alignment of NEs and right-branching NPs, the increase in performance was negligible.

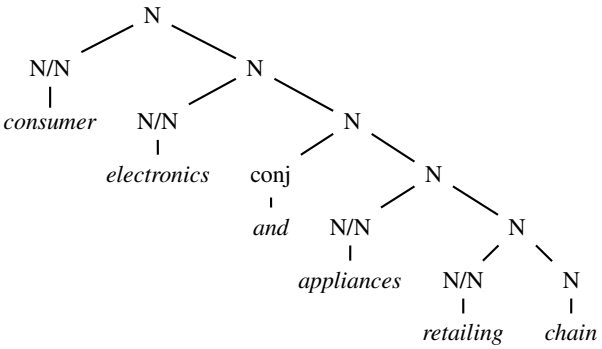


Figure 1: CCG derivation from Hockenmaier (2003)

3 Background

The NP bracketing task has often been posed in terms of choosing between the left or right branching structure of three word noun compounds:

(a) *(world (oil prices))* – Right-branching

(b) *((crude oil) prices)* – Left-branching

Most approaches to the problem use unsupervised methods, based on competing association strength between two of the words in the compound (Marcus, 1980, p. 253). There are two possible models to choose from: dependency or adjacency. The *dependency model* compares the association between words 1-2 to words 1-3, while the *adjacency model* compares words 1-2 to words 2-3.

Lauer (1995) has demonstrated superior performance of the dependency model using a test set of 244 (216 unique) noun compounds drawn from Grolier's encyclopedia. This data has been used to evaluate most research since. He uses Roget's thesaurus to smooth words into semantic classes, and then calculates association between classes based on their counts in a "training set" also drawn from Grolier's. He achieves 80.7% accuracy using POS tags to indentify bigrams in the training set.

Lapata and Keller (2004) derive estimates from web counts, and only compare at a lexical level, achieving 78.7% accuracy. Nakov and Hearst (2005) also use web counts, but incorporate additional counts from several variations on simple bigram queries, including queries for the pairs of words concatenated or joined by a hyphen. This results in an impressive 89.3% accuracy.

There have also been attempts to solve this task using supervised methods, even though the lack of gold-standard data makes this difficult. Girju et al.

(2005) draw a training set from raw WSJ text and use it to train a decision tree classifier achieving 73.1% accuracy. When they shuffled their data with Lauer's to create a new test and training split, their accuracy increased to 83.1% which may be a result of the ∼10% duplication in Lauer's test set.

We have created a new NP bracketing data set from our extended Treebank by extracting all rightmost three noun sequences from base-NPs. Our initial experiments are presented in Section 6.1.

4 Corpus Creation

According to Marcus et al. (1993), asking annotators to markup base-NP structure significantly reduced annotation speed, and for this reason base-NPs were left flat. The bracketing guidelines (Bies et al., 1995) also mention the considerable difficulty of identifying the correct scope for nominal modifiers. We found however, that while there are certainly difficult cases, the vast majority are quite simple and can be annotated reliably. Our annotation philosophy can be summarised as:

1. most cases are easy and fit a common pattern;
2. prefer the implicit right-branching structure for difficult decisions. Finance jargon was a common source of these;
3. mark very difficult to bracket NPs and discuss with other annotators later;

During this process we identified numerous cases that require a more sophisticated annotation scheme. There are genuine flat cases, primarily names like *John A. Smith*, that we would like to distinguish from implicitly right-branching NPs in the next version of the corpus. Although our scheme is still developing, we believe that the current annotation is already useful for statistical modelling, and we demonstrate this empirically in Section 6.

4.1 Annotation Process

Our annotation guidelines[1] are based on those developed for annotating full sub-NP structure in the biomedical domain (Kulick et al., 2004). The annotation guidelines for this biomedical corpus (an addendum to the Penn Treebank guidelines) introduce the use of NML nodes to mark internal NP structure.

In summary, our guidelines leave right-branching structures untouched, and insert labelled brackets around left-branching structures. The label of the newly created constituent is NML or JJP, depending on whether its head is a noun or an adjective. We also chose not to alter the existing Penn Treebank annotation, even though the annotators found many errors during the annotation process. We wanted to keep our extended Treebank as similar to the original as possible, so that they remain comparable.

We developed a bracketing tool, which identifies ambiguous NPs and presents them to the user for disambiguation. An ambiguous NP is any (possibly non-base) NP with three or more contiguous children that are either single words or another NP. Certain common patterns, such as three words beginning with a determiner, are unambiguous, and were filtered out. The annotator is also shown the entire sentence surrounding the ambiguous NP.

The bracketing tool often suggests a bracketing using rules based mostly on named entity tags, which are drawn from the BBN corpus (Weischedel and Brunstein, 2005). For example, since *Air Force* is given ORG tags, the tool suggests that they be bracketed together first. Other suggestions come from previous bracketings of the same words, which helps to keep the annotator consistent.

Two post processes were carried out to increase annotation consistency and correctness. 915 difficult NPs were marked by the annotator and were then discussed with two other experts. Secondly, certain phrases that occurred numerous times and were non-trivial to bracket, e.g. *London Interbank Offered Rate*, were identified. An extra pass was made through the corpus, ensuring that every instance of these phrases was bracketed consistently.

4.2 Annotation Time

Annotation initially took over 9 hours per section of the Treebank. However, with practice this was reduced to about 3 hours per section. Each section contains around 2500 ambiguous NPs, i.e. annotating took approximately 5 seconds per NP. Most NPs require no bracketing, or fit into a standard pattern which the annotator soon becomes accustomed to, hence the task can be performed quite quickly.

For the original bracketing of the Treebank, annotators performed at 375–475 words per hour after a

[1] The guidelines and corpus are available on our webpages.

	PREC.	RECALL	F-SCORE
Brackets	89.17	87.50	88.33
Dependencies	96.40	96.40	96.40
Brackets, revised	97.56	98.03	97.79
Dependencies, revised	99.27	99.27	99.27

Table 1: Agreement between annotators

few weeks, and increased to about 1000 words per hour after gaining more experience (Marcus et al., 1993). For our annotation process, counting each word in every NP shown, our speed was around 800 words per hour. This figure is not unexpected, as the task was not large enough to get more than a month's experience, and there is less structure to annotate.

5 Corpus Analysis

5.1 Inter-annotator Agreement

The annotation was performed by the first author. A second Computational Linguistics PhD student also annotated Section 23, allowing inter-annotator agreement, and the reliability of the annotations, to be measured. This also maximised the quality of the section used for parser testing.

We measured the proportion of matching brackets and dependencies between annotators, shown in Table 1, both before and after they discussed cases of disagreement and revised their annotations. The number of dependencies is fixed by the length of the NP, so the dependency precision and recall are the same. Counting matched brackets is a harsher evaluation, as there are many NPs that both annotators agree should have no additional bracketing, which are not taken into account by the metric.

The disagreements occurred for a small number of repeated instances, such as this case:

```
(NP                          (NP (NNP Goldman)
  (NML (NNP Goldman)           (, ,)
    (, ,)                      (NNP Sachs)
    (NNP Sachs) )              (CC &) (NNP Co) )
  (CC &) (NNP Co) )
```

The first annotator felt that `Goldman , Sachs` should form their own NML constituent, while the second annotator did not.

We can also look at exact matching on NPs, where the annotators originally agreed in 2667 of 2908 cases (91.71%), and after revision, in 2864 of 2907 cases (98.52%). These results demonstrate that high agreement rates are achievable for these annotations.

	MATCHED	TOTAL	%
By dependency	1409 (1315)	1479	95.27 (88.91)
By noun phrase	562 (489)	626	89.78 (78.12)
By dependency, only annotated NPs	578 (543)	627	92.19 (86.60)
By noun phrase, only annotated NPs	186 (162)	229	81.22 (70.74)

Table 2: Agreement with DepBank

5.2 DepBank Agreement

Another approach to measuring annotator reliability is to compare with an independently annotated corpus on the same text. We used the PARC700 Dependency Bank (King et al., 2003) which consists of 700 Section 23 sentences annotated with labelled dependencies. We use the Briscoe and Carroll (2006) version of DepBank, a 560 sentence subset used to evaluate the RASP parser.

Some translation is required to compare our brackets to DepBank dependencies. We map the brackets to dependencies by finding the head of the NP, using the Collins (1999) head finding rules, and then creating a dependency between each other child's head and this head. This does not work perfectly, and mismatches occur because of which dependencies DepBank marks explicitly, and how it chooses heads. The errors are investigated manually to determine their cause.

The results are shown in Table 2, with the number of agreements before manual checking shown in parentheses. Once again the dependency numbers are higher than those at the NP level. Similarly, when we only look at cases where we have inserted some annotations, we are looking at more difficult cases and the score is not as high.

The results of this analysis are quite positive. Over half of the disagreements that occur (in either measure) are caused by how company names are bracketed. While we have always separated the company name from post-modifiers such as *Corp* and *Inc*, DepBank does not in most cases. These results show that consistently and correctly bracketing noun phrase structure is possible, and that inter-annotator agreement is at an acceptable level.

5.3 Corpus Composition and Consistency

Looking at the entire Penn Treebank corpus, the annotation tool finds 60959 ambiguous NPs out of the 432639 NPs in the corpus (14.09%). 22851 of

LEVEL	COUNT	POS TAGS	EXAMPLE
NP	1073	JJ JJ NNS	big red cars
	1581	DT JJ NN NN	a high interest rate
	1693	JJ NN NNS	high interest rates
	3557	NNP NNP NNP	John A. Smith
NML	1468	NN NN	(interest rate) rises
	1538	JJ NN	(heavy truck) rentals
	1650	NNP NNP NNP	(A. B. C.) Corp
	8524	NNP NNP	(John Smith) Jr.
JJP	82	JJ JJ	(dark red) car
	114	RB JJ	(very high) rates
	122	JJ CC JJ	(big and red) apples
	160	" JJ "	(" smart ") cars

Table 3: Common POS tag sequences

these (37.49%) had brackets inserted by the annotator. This is as we expect, as the majority of NPs are right-branching. Of the brackets added, 22368 were NML nodes, while 863 were JJP.

To compare, we can count the number of existing NP and ADJP nodes found in the NPs that the bracketing tool presents. We find there are 32772 NP children, and 579 ADJP, which are quite similar numbers to the amount of nodes we have added. From this, we can say that our annotation process has introduced almost as much structural information into NPs as there was in the original Penn Treebank.

Table 3 shows the most common POS tag sequences for NP, NML and JJP nodes. An example is given showing typical words that match the POS tags. For NML and JJP, we also show the words bracketed, as they would appear under an NP node.

We checked the consistency of the annotations by identifying NPs with the same word sequence and checking whether they were always bracketed identically. After the first pass through, there were 360 word sequences with multiple bracketings, which occurred in 1923 NP instances. 489 of these instances differed from the majority case for that sequence, and were probably errors.

The annotator had marked certain difficult and commonly repeating NPs. From this we generated a list of phrases, and then made another pass through the corpus, synchronising all instances that contained one of these phrases. After this, only 150 instances differed from the majority case. Inspecting these remaining inconsistencies showed cases like:

```
(NP-TMP (NML (NNP Nov.) (CD 15))
  (, ,)
  (CD 1999))
```

where we were inconsistent in inserting the NML node because the Penn Treebank sometimes already has the structure annotated under an NP node. Since we do not make changes to existing brackets, we cannot fix these cases. Other inconsistencies are rare, but will be examined and corrected in a future release.

The annotator made a second pass over Section 00 to correct changes made after the beginning of the annotation process. Comparing the two passes can give us some idea of how the annotator changed as he grew more practiced at the task.

We find that the old and new versions are identical in 88.65% of NPs, with labelled precision, recall and F-score being 97.17%, 76.69% and 85.72% respectively. This tells us that there were many brackets originally missed that were added in the second pass. This is not surprising since the main problem with how Section 00 was annotated originally was that company names were not separated from their post-modifier (such as *Corp*). Besides this, it suggests that there is not a great deal of difference between an annotator just learning the task, and one who has had a great deal of experience with it.

5.4 Named Entity Suggestions

We have also evaluated how well the suggestion feature of the annotation tool performs. In particular, we want to determine how useful named entities are in determining the correct bracketing.

We ran the tool over the original corpus, following NE-based suggestions where possible. We find that when evaluated against our annotations, the F-score is 50.71%. We need to look closer at the precision and recall though, as they are quite different. The precision of 93.84% is quite high. However, there are many brackets where the entities do not help at all, and so the recall of this method was only 34.74%. This suggests that a NE feature may help to identify the correct bracketing in one third of cases.

6 Experiments

Having bracketed NPs in the Penn Treebank, we now describe our initial experiments on how this additional level of annotation can be exploited.

6.1 NP Bracketing Data

The obvious first task to consider is noun phrase bracketing itself. We implement a similar system to

CORPUS	# ITEMS	LEFT	RIGHT
Penn Treebank	5582	58.99%	41.01%
Lauer's	244	66.80%	33.20%

Table 4: Comparison of NP bracketing corpora

N-GRAM	MATCH
Unigrams	51.20%
Adjacency bigrams	6.35%
Dependency bigrams	3.85%
All bigrams	5.83%
Trigrams	1.40%

Table 5: Lexical overlap

Lauer (1995), described in Section 3, and report on results from our own data and Lauer's original set.

First, we extracted three word noun sequences from all the ambiguous NPs. If the last three children are nouns, then they became an example in our data set. If there is a NML node containing the first two nouns then it is left-branching, otherwise it is right-branching. Because we are only looking at the right-most part of the NP, we know that we are not extracting any nonsensical items. We also remove all items where the nouns are all part of a named entity to eliminate flat structure cases.

Statistics about the new data set and Lauer's data set are given in Table 4. As can be seen, the Penn Treebank based corpus is significantly larger, and has a more even mix of left and right-branching noun phrases. We also measured the amount of lexical overlap between the two corpora, shown in Table 5. This displays the percentage of n-grams in Lauer's corpus that are also in our corpus. We can clearly see that the two corpora are quite dissimilar, as even on unigrams barely half are shared.

6.2 NP Bracketing Results

With our new data set, we began running experiments similar to those carried out in the literature (Nakov and Hearst, 2005). We implemented both an adjacency and dependency model, and three different association measures: raw counts, bigram probability, and χ^2. We draw our counts from a corpus of n-gram counts calculated over 1 trillion words from the web (Brants and Franz, 2006).

The results from the experiments, on both our and Lauer's data set, are shown in Table 6. Our results

ASSOC. MEASURE	LAUER	PTB
Raw counts, adj.	75.41%	77.46%
Raw counts, dep.	77.05%	68.85%
Probability, adj.	71.31%	76.42%
Probability, dep.	**80.33%**	69.56%
χ^2, adj.	71.31%	**77.93%**
χ^2, dep.	74.59%	68.92%

Table 6: Bracketing task, unsupervised results

FEATURES	LAUER	10-FOLD CROSS
All features	80.74%	**89.91%** (1.04%)
Lexical	71.31%	**84.52%** (1.77%)
n-gram counts	75.41%	82.50% (1.49%)
Probability	72.54%	78.34% (2.11%)
χ^2	75.41%	80.10% (1.71%)
Adjacency model	72.95%	79.52% (1.32%)
Dependency model	**78.69%**	72.86% (1.48%)
Both models	76.23%	79.67% (1.42%)
-Lexical	79.92%	85.72% (0.77%)
-n-gram counts	80.74%	89.11% (1.39%)
-Probability	79.10%	**89.79%** (1.22%)
-χ^2	80.74%	**89.79%** (0.98%)
-Adjacency model	81.56%	89.63% (0.96%)
-Dependency model	81.15%	89.72% (0.86%)
-Both models	**81.97%**	89.63% (0.95%)

Table 7: Bracketing task, supervised results

on Lauer's corpus are similar to those reported previously, with the dependency model outperforming the adjacency model on all measures. The bigram probability scores highest out of all the measures, while the χ^2 score performed the worst.

The results on the new corpus are even more surprising, with the adjacency model outperforming the dependency model by a wide margin. The χ^2 measure gives the highest accuracy, but still only just outperforms the raw counts. Our analysis shows that the good performance of the adjacency model comes from the large number of named entities in the corpus. When we remove all items that have any word as an entity, the results change, and the dependency model is superior. We also suspect that another cause of the unusual results is the different proportions of left and right-branching NPs.

With a large annotated corpus, we can now run supervised NP bracketing experiments. We present two configurations in Table 7: training on our corpus and testing on Lauer's set; and performing 10-fold cross validation using our corpus alone.

The feature set we explore encodes the information we used in the unsupervised experiments. Ta-

	OVERALL			ONLY NML JJP			NOT NML JJP		
	PREC.	RECALL	F-SCORE	PREC.	RECALL	F-SCORE	PREC.	RECALL	F-SCORE
Original	88.93	88.90	88.92	–	–	–	88.93	88.90	88.92
NML and JJP bracketed	88.63	88.29	88.46	77.93	62.93	69.63	88.85	88.93	88.89
Relabelled brackets	88.17	87.88	88.02	91.93	51.38	65.91	87.86	88.65	88.25

Table 8: Parsing performance

ble 7 shows the performance with: all features, followed by the individual features, and finally, after removing individual features.

The feature set includes: lexical features for each n-gram in the noun compound; n-gram counts for unigrams, bigrams and trigrams; raw probability and χ^2 association scores for all three bigrams in the compound; and the adjacency and dependency results for all three association measures. We discretised the non-binary features using an implementation of Fayyad and Irani's (1993) algorithm, and classify using MegaM[2].

The results on Lauer's set demonstrate that the dependency model performs well by itself but not with the other features. In fact, a better result comes from using every feature *except* those from the dependency and adjacency models. It is also impressive how good the performance is, considering the large differences between our data set and Lauer's.

These differences also account for the disparate cross-validation figures. On this data, the lexical features perform the best, which is to be expected given the nature of the corpus. The best model in this case comes from using all the features.

6.3 Collins Parsing

We can also look at the impact of our new annotations upon full statistical parsing. We use Bikel's implementation (Bikel, 2004) of Collins' parser (Collins, 1999) in order to carry out these experiments, using the non-deficient Collins settings. The numbers we give are labelled bracket precision, recall and F-scores for all sentences. Bikel mentions that base-NPs are treated very differently in Collins' parser, and so it will be interesting to observe the results using our new annotations.

Firstly, we compare the parser's performance on the original Penn Treebank and the new NML and JJP bracketed version. Table 8 shows that the new brackets make parsing marginally more difficult overall (by about 0.5% in F-score).

The performance on only the new NML and JJP brackets is not very high. This shows the difficulty of correctly bracketing NPs. Conversely, the figures for all brackets except NML and JJP are only a tiny amount less in our extended corpus. This means that performance for other phrases is hardly changed by the new NP brackets.

We also ran an experiment where the new NML and JJP labels were relabelled as NP and ADJP. These are the labels that would be given if NPs were originally bracketed with the rest of the Penn Treebank. This meant the model would not have to discriminate between two different types of noun and adjective structure. The performance, as shown in Table 8, was even lower with this approach, suggesting that the distinction is larger than we anticipated. On the other hand, the precision on NML and JJP constituents was quite high, so the parser is able to identify at least some of the structure very well.

7 Conclusion

The work presented in this paper is a first step towards accurate representation of noun phrase structure in NLP corpora. There are several distinctions that our annotation currently ignores that we would like to identify correctly in the future. Firstly, NPs with genuine flat structure are currently treated as implicitly right branching. Secondly, there is still ambiguity in determining the head of a noun phrase. Although Collins' head finding rules work in most NPs, there are cases such as *IBM Australia* where the head is not the right-most noun. Similarly, apposition is very common in the Penn Treebank, in NPs such as *John Smith , IBM president*. We would like to be able to identify these multi-head constructs properly, rather than simply treating them as a single entity (or even worse, as two different entities).

Having the correct NP structure also means that we can now represent the true structure in CCGbank, one of the problems we described earlier. Transfer-

[2]Available at http://www.cs.utah.edu/ hal/megam/

ring our annotations should be fairly simple, requiring just a few changes to how NPs are treated in the current translation process.

The addition of consistent, gold-standard, noun phrase structure to a large corpus is a significant achievement. We have shown that the these annotations can be created in a feasible time frame with high inter-annotator agreement of 98.52% (measuring exact NP matches). The new brackets cause only a small drop in parsing performance, and no significant decrease on the existing structure. As NEs were useful for suggesting brackets automatically, we intend to incorporate NE information into statistical parsing models in the future.

Our annotated corpus can improve the performance of any system that relies on NPs from parsers trained on the Penn Treebank. A Collins' parser trained on our corpus is now able to identify sub-NP brackets, making it of use in other NLP systems. QA systems, for example, will be able to exploit internal NP structure. In the future, we will improve the parser's performance on NML and JJP brackets.

We have provided a significantly larger corpus for analysing NP structure than has ever been made available before. This is integrated within perhaps the most influential corpus in NLP. The large number of systems trained on Penn Treebank data can all benefit from the extended resource we have created.

Acknowledgements

We would like to thank Matthew Honnibal, our second annotator, who also helped design the guidelines; Toby Hawker, for implementing the discretiser; Mark Lauer for releasing his data; and the anonymous reviewers for their helpful feedback. This work has been supported by the Australian Research Council under Discovery Projects DP0453131 and DP0665973.

References

Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. 1995. Bracketing guidelines for Treebank II style Penn Treebank project. Technical report, University of Pennsylvania.

Dan Bikel. 2004. *On the Parameter Space of Generative Lexicalized Statistical Parsing Models*. Ph.D. thesis, University of Pennsylvania.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram version 1. Linguistic Data Consortium.

Ted Briscoe and John Carroll. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the PARC DepBank. In *Proceedings of the Poster Session of COLING/ACL-06*. Sydney, Australia.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Usama M. Fayyad and Keki B. Irani. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artifical Intelligence (IJCAI–93)*, pages 1022–1029. Chambery, France.

Roxana Girju, Dan Moldovan, Marta Tatu, and Daniel Antohe. 2005. On the semantics of noun compounds. *Journal of Computer Speech and Language - Special Issue on Multiword Expressions*, 19(4):313–330.

Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Tracy Holloway King, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The PARC700 dependency bank. In *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*. Budapest, Hungary.

Seth Kulick, Ann Bies, Mark Libeman, Mark Mandel, Ryan McDonald, Martha Palmer, Andrew Schein, and Lyle Ungar. 2004. Integrated annotation for biomedical information extraction. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. Boston.

Mirella Lapata and Frank Keller. 2004. The web as a baseline: Evaluating the performance of unsupervised web-based models for a range of NLP tasks. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 121–128. Boston.

Mark Lauer. 1995. Corpus statistics meet the compound noun: Some empirical results. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*. Cambridge, MA.

Mitchell Marcus. 1980. *A Theory of Syntactic Recognition for Natural Language*. MIT Press, Cambridge, MA.

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

Preslav Nakov and Marti Hearst. 2005. Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *Proceedings of CoNLL-2005, Ninth Conference on Computational Natural Language Learning*. Ann Arbor, MI.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*. Cambridge MA, USA.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA.

Ralph Weischedel and Ada Brunstein. 2005. BBN pronoun coreference and entity type corpus. Technical report, Linguistic Data Consortium.

Formalism-Independent Parser Evaluation with CCG and DepBank

Stephen Clark
Oxford University Computing Laboratory
Wolfson Building, Parks Road
Oxford, OX1 3QD, UK
stephen.clark@comlab.ox.ac.uk

James R. Curran
School of Information Technologies
University of Sydney
NSW 2006, Australia
james@it.usyd.edu.au

Abstract

A key question facing the parsing community is how to compare parsers which use different grammar formalisms and produce different output. Evaluating a parser on the same resource used to create it can lead to non-comparable accuracy scores and an over-optimistic view of parser performance. In this paper we evaluate a CCG parser on DepBank, and demonstrate the difficulties in converting the parser output into DepBank grammatical relations. In addition we present a method for measuring the effectiveness of the conversion, which provides an upper bound on parsing accuracy. The CCG parser obtains an F-score of 81.9% on labelled dependencies, against an upper bound of 84.8%. We compare the CCG parser against the RASP parser, outperforming RASP by over 5% overall and on the majority of dependency types.

1 Introduction

Parsers have been developed for a variety of grammar formalisms, for example HPSG (Toutanova et al., 2002; Malouf and van Noord, 2004), LFG (Kaplan et al., 2004; Cahill et al., 2004), TAG (Sarkar and Joshi, 2003), CCG (Hockenmaier and Steedman, 2002; Clark and Curran, 2004b), and variants of phrase-structure grammar (Briscoe et al., 2006), including the phrase-structure grammar implicit in the Penn Treebank (Collins, 2003; Charniak, 2000). Different parsers produce different output, for example phrase structure trees (Collins, 2003), dependency trees (Nivre and Scholz, 2004), grammatical relations (Briscoe et al., 2006), and formalism-specific dependencies (Clark and Curran, 2004b). This variety of formalisms and output creates a challenge for parser evaluation.

The majority of parser evaluations have used test sets drawn from the same resource used to develop the parser. This allows the many parsers based on the Penn Treebank, for example, to be meaningfully compared. However, there are two drawbacks to this approach. First, parser evaluations using different resources cannot be compared; for example, the Parseval scores obtained by Penn Treebank parsers cannot be compared with the dependency F-scores obtained by evaluating on the Parc Dependency Bank. Second, using the same resource for development and testing can lead to an over-optimistic view of parser performance.

In this paper we evaluate a CCG parser (Clark and Curran, 2004b) on the Briscoe and Carroll version of DepBank (Briscoe and Carroll, 2006). The CCG parser produces head-dependency relations, so evaluating the parser should simply be a matter of converting the CCG dependencies into those in DepBank. Such conversions have been performed for other parsers, including parsers producing phrase structure output (Kaplan et al., 2004; Preiss, 2003). However, we found that performing such a conversion is a time-consuming and non-trivial task.

The contributions of this paper are as follows. First, we demonstrate the considerable difficulties associated with formalism-independent parser evaluation, highlighting the problems in converting the

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 248–255,
Prague, Czech Republic, June 2007.

output of a parser from one representation to another. Second, we develop a method for measuring how effective the conversion process is, which also provides an upper bound for the performance of the parser, given the conversion process being used; this method can be adapted by other researchers to strengthen their own parser comparisons. And third, we provide the first evaluation of a wide-coverage CCG parser outside of CCGbank, obtaining impressive results on DepBank and outperforming the RASP parser (Briscoe et al., 2006) by over 5% overall and on the majority of dependency types.

2 Previous Work

The most common form of parser evaluation is to apply the Parseval metrics to phrase-structure parsers based on the Penn Treebank, and the highest reported scores are now over 90% (Bod, 2003; Charniak and Johnson, 2005). However, it is unclear whether these high scores accurately reflect the performance of parsers in applications. It has been argued that the Parseval metrics are too forgiving and that phrase structure is not the ideal representation for a gold standard (Carroll et al., 1998). Also, using the same resource for training and testing may result in the parser learning systematic errors which are present in both the training and testing material. An example of this is from CCGbank (Hockenmaier, 2003), where all modifiers in noun-noun compound constructions modify the final noun (because the Penn Treebank, from which CCGbank is derived, does not contain the necessary information to obtain the correct bracketing). Thus there are non-negligible, systematic errors in both the training and testing material, and the CCG parsers are being rewarded for following particular mistakes.

There are parser evaluation suites which have been designed to be formalism-independent and which have been carefully and manually corrected. Carroll et al. (1998) describe such a suite, consisting of sentences taken from the Susanne corpus, annotated with Grammatical Relations (GRs) which specify the syntactic relation between a head and dependent. Thus all that is required to use such a scheme, in theory, is that the parser being evaluated is able to identify heads. A similar resource — the Parc Dependency Bank (DepBank) (King et al., 2003) — has been created using sentences from the Penn Treebank. Briscoe and Carroll (2006) reannotated this resource using their GRs scheme, and used it to evaluate the RASP parser.

Kaplan et al. (2004) compare the Collins (2003) parser with the Parc LFG parser by mapping LFG F-structures and Penn Treebank parses into DepBank dependencies, claiming that the LFG parser is considerably more accurate with only a slight reduction in speed. Preiss (2003) compares the parsers of Collins (2003) and Charniak (2000), the GR finder of Buchholz et al. (1999), and the RASP parser, using the Carroll et al. (1998) gold-standard. The Penn Treebank trees of the Collins and Charniak parsers, and the GRs of the Buchholz parser, are mapped into the required GRs, with the result that the GR finder of Buchholz is the most accurate.

The major weakness of these evaluations is that there is no measure of the difficultly of the conversion process for each of the parsers. Kaplan et al. (2004) clearly invested considerable time and expertise in mapping the output of the Collins parser into the DepBank dependencies, but they also note that "This conversion was relatively straightforward for LFG structures . . . However, a certain amount of skill and intuition was required to provide a fair conversion of the Collins trees". Without some measure of the difficulty — and effectiveness — of the conversion, there remains a suspicion that the Collins parser is being unfairly penalised.

One way of providing such a measure is to convert the original gold standard on which the parser is based and evaluate that against the new gold standard (assuming the two resources are based on the same corpus). In the case of Kaplan et al. (2004), the testing procedure would include running their conversion process on Section 23 of the Penn Treebank and evaluating the output against DepBank. As well as providing some measure of the effectiveness of the conversion, this method would also provide an upper bound for the Collins parser, giving the score that a perfect Penn Treebank parser would obtain on DepBank (given the conversion process).

We perform such an evaluation for the CCG parser, with the surprising result that the upper bound on DepBank is only 84.8%, despite the considerable effort invested in developing the conversion process.

3 The CCG Parser

Clark and Curran (2004b) describes the CCG parser used for the evaluation. The grammar used by the parser is extracted from CCGbank, a CCG version of the Penn Treebank (Hockenmaier, 2003). The grammar consists of 425 lexical categories — expressing subcategorisation information — plus a small number of combinatory rules which combine the categories (Steedman, 2000). A supertagger first assigns lexical categories to the words in a sentence, which are then combined by the parser using the combinatory rules and the CKY algorithm. A log-linear model scores the alternative parses. We use the normal-form model, which assigns probabilities to single derivations based on the normal-form derivations in CCGbank. The features in the model are defined over local parts of the derivation and include word-word dependencies. A packed chart representation allows efficient decoding, with the Viterbi algorithm finding the most probable derivation.

The parser outputs predicate-argument dependencies defined in terms of CCG lexical categories. More formally, a CCG predicate-argument dependency is a 5-tuple: $\langle h_f, f, s, h_a, l \rangle$, where h_f is the lexical item of the lexical category expressing the dependency relation; f is the lexical category; s is the argument slot; h_a is the head word of the argument; and l encodes whether the dependency is long-range. For example, the dependency encoding *company* as the object of *bought* (as in *IBM bought the company*) is represented as follows:

$$\langle \textit{bought}, (S\backslash NP_1)/NP_2, 2, \textit{company}, - \rangle \quad (1)$$

The lexical category $(S\backslash NP_1)/NP_2$ is the category of a transitive verb, with the first argument slot corresponding to the subject, and the second argument slot corresponding to the direct object. The final field indicates the nature of any long-range dependency; in (1) the dependency is local.

The predicate-argument dependencies — including long-range dependencies — are encoded in the lexicon by adding head and dependency annotation to the lexical categories. For example, the expanded category for the control verb *persuade* is $(((S[dcl]_{persuade}\backslash NP_1)/(S[to]_2\backslash NP_X))/NP_{X,3})$. Numerical subscripts on the argument categories represent dependency relations; the head of the final declarative sentence is *persuade*; and the head of the infinitival complement's subject is identified with the head of the object, using the variable *X*, as in standard unification-based accounts of control.

Previous evaluations of CCG parsers have used the predicate-argument dependencies from CCGbank as a test set (Hockenmaier and Steedman, 2002; Clark and Curran, 2004b), with impressive results of over 84% F-score on labelled dependencies. In this paper we reinforce the earlier results with the first evaluation of a CCG parser outside of CCGbank.

4 Dependency Conversion to DepBank

For the gold standard we chose the version of DepBank reannotated by Briscoe and Carroll (2006), consisting of 700 sentences from Section 23 of the Penn Treebank. The B&C scheme is similar to the original DepBank scheme (King et al., 2003), but overall contains less grammatical detail; Briscoe and Carroll (2006) describes the differences. We chose this resource for the following reasons: it is publicly available, allowing other researchers to compare against our results; the GRs making up the annotation share some similarities with the predicate-argument dependencies output by the CCG parser; and we can directly compare our parser against a non-CCG parser, namely the RASP parser. We chose not to use the corpus based on the Susanne corpus (Carroll et al., 1998) because the GRs are less like the CCG dependencies; the corpus is not based on the Penn Treebank, making comparison more difficult because of tokenisation differences, for example; and the latest results for RASP are on DepBank.

The GRs are described in Briscoe and Carroll (2006) and Briscoe et al. (2006). Table 1 lists the GRs used in the evaluation. As an example, the sentence *The parent sold Imperial* produces three GRs: `(det parent The)`, `(ncsubj sold parent _)` and `(dobj sold Imperial)`. Note that some GRs — in this example `ncsubj` — have a *subtype slot*, giving extra information. The subtype slot for `ncsubj` is used to indicate passive subjects, with the null value "_" for active subjects and `obj` for passive subjects. Other subtype slots are discussed in Section 4.2.

The CCG dependencies were transformed into GRs in two stages. The first stage was to create a mapping between the CCG dependencies and the

GR	description
conj	coordinator
aux	auxiliary
det	determiner
ncmod	non-clausal modifier
xmod	unsaturated predicative modifier
cmod	saturated clausal modifier
pmod	PP modifier with a PP complement
ncsubj	non-clausal subject
xsubj	unsaturated predicative subject
csubj	saturated clausal subject
dobj	direct object
obj2	second object
iobj	indirect object
pcomp	PP which is a PP complement
xcomp	unsaturated VP complement
ccomp	saturated clausal complement
ta	textual adjunct delimited by punctuation

Table 1: GRs in B&C's annotation of DepBank

GRs. This involved mapping each argument slot in the 425 lexical categories in the CCG lexicon onto a GR. In the second stage, the GRs created from the parser output were post-processed to correct some of the obvious remaining differences between the CCG and GR representations.

In the process of performing the transformation we encountered a methodological problem: without looking at examples it was difficult to create the mapping and impossible to know whether the two representations were converging. Briscoe et al. (2006) split the 700 sentences in DepBank into a test and development set, but the latter only consists of 140 sentences which was not enough to reliably create the transformation. There are some development files in the RASP release which provide examples of the GRs, which were used when possible, but these only cover a subset of the CCG lexical categories.

Our solution to this problem was to convert the gold standard dependencies from CCGbank into GRs and use these to develop the transformation. So we did inspect the annotation in DepBank, and compared it to the transformed CCG dependencies, but only the *gold-standard* CCG dependencies. Thus the parser output was never used during this process. We also ensured that the dependency mapping and the post processing are general to the GRs scheme and not specific to the test set or parser.

4.1 Mapping the CCG dependencies to GRs

Table 2 gives some examples of the mapping; `%l` indicates the word associated with the lexical category

CCG lexical category	slot	GR
$(S[dcl]\backslash NP_1)/NP_2$	1	`(ncsubj %l %f _)`
$(S[dcl]\backslash NP_1)/NP_2$	2	`(dobj %l %f)`
$(S\backslash NP)/(S\backslash NP)_1$	1	`(ncmod _ %f %l)`
$(NP\backslash NP_1)/NP_2$	1	`(ncmod _ %f %l)`
$(NP\backslash NP_1)/NP_2$	2	`(dobj %l %f)`
$NP[nb]/N_1$	1	`(det %f %l)`
$(NP\backslash NP_1)/(S[pss]\backslash NP)_2$	1	`(xmod _ %f %l)`
$(NP\backslash NP_1)/(S[pss]\backslash NP)_2$	2	`(xcomp _ %l %f)`
$((S\backslash NP)\backslash(S\backslash NP)_1)/S[dcl]_2$	1	`(cmod _ %f %l)`
$((S\backslash NP)\backslash(S\backslash NP)_1)/S[dcl]_2$	2	`(ccomp _ %l %f)`
$((S[dcl]\backslash NP_1)/NP_2)/NP_3$	2	`(obj2 %l %f)`
$(S[dcl]\backslash NP_1)/(S[b]\backslash NP)_2$	2	`(aux %f %l)`

Table 2: Examples of the dependency mapping

and `%f` is the head of the constituent filling the argument slot. Note that the order of `%l` and `%f` varies according to whether the GR represents a complement or modifier, in line with the Briscoe and Carroll annotation. For many of the CCG dependencies, the mapping into GRs is straightforward. For example, the first two rows of Table 2 show the mapping for the transitive verb category $(S[dcl]\backslash NP_1)/NP_2$: argument slot 1 is a non-clausal subject and argument slot 2 is a direct object.

Creating the dependency transformation is more difficult than these examples suggest. The first problem is that the mapping from CCG dependencies to GRs is many-to-many. For example, the transitive verb category $(S[dcl]\backslash NP)/NP$ applies to the copula in sentences like *Imperial Corp. is the parent of Imperial Savings & Loan*. With the default annotation, the relation between *is* and *parent* would be `dobj`, whereas in DepBank the argument of the copula is analysed as an `xcomp`. Table 3 gives some examples of how we attempt to deal with this problem. The constraint in the first example means that, whenever the word associated with the transitive verb category is a form of *be*, the second argument is `xcomp`, otherwise the default case applies (in this case `dobj`). There are a number of categories with similar constraints, checking whether the word associated with the category is a form of *be*.

The second type of constraint, shown in the third line of the table, checks the lexical category of the word filling the argument slot. In this example, if the lexical category of the preposition is PP/NP, then the second argument of $(S[dcl]\backslash NP)/PP$ maps to `iobj`; thus in *The loss stems from several factors* the relation between the verb and preposition is `(iobj stems from)`. If the lexical category of

CCG lexical category	slot	GR	constraint	example
$(S[dcl]\backslash NP_1)/NP_2$	2	(xcomp _ %l %f)	word=be	The parent *is Imperial*
		(dobj %l %f)		The parent *sold Imperial*
$(S[dcl]\backslash NP_1)/PP_2$	2	(iobj %l %f)	cat=PP/NP	The loss *stems from* several factors
		(xcomp _ %l %f)	cat=$PP/(S[ng]\backslash NP)$	The future *depends on* building ties
$(S[dcl]\backslash NP_1)/(S[to]\backslash NP)_2$	2	(xcomp %f %l %k)	cat=$(S[to]\backslash NP)/(S[b]\backslash NP)$	*wants to wean* itself away from

Table 3: Examples of the many-to-many nature of the CCG dependency to GRs mapping, and a ternary GR

the preposition is $PP/(S[ng]\backslash NP)$, then the GR is xcomp; thus in *The future depends on building ties* the relation between the verb and preposition is (xcomp _ depends on). There are a number of CCG dependencies with similar constraints, many of them covering the iobj/xcomp distinction.

The second difficulty is that not all the GRs are binary relations, whereas the CCG dependencies are all binary. The primary example of this is to-infinitival constructions. For example, in the sentence *The company wants to wean itself away from expensive gimmicks*, the CCG parser produces two dependencies relating *wants*, *to* and *wean*, whereas there is only one GR: (xcomp to wants wean). The final row of Table 3 gives an example. We implement this constraint by introducing a %k variable into the GR template which denotes the argument of the category in the constraint column (which, as before, is the lexical category of the word filling the argument slot). In the example, the current category is $(S[dcl]\backslash NP_1)/(S[to]\backslash NP)_2$, which is associated with *wants*; this combines with $(S[to]\backslash NP)/(S[b]\backslash NP)$, associated with *to*; and the argument of $(S[to]\backslash NP)/(S[b]\backslash NP)$ is *wean*. The %k variable allows us to look beyond the arguments of the current category when creating the GRs.

A further difficulty is that the head passing conventions differ between DepBank and CCGbank. By *head passing* we mean the mechanism which determines the heads of constituents and the mechanism by which words become arguments of long-range dependencies. For example, in the sentence *The group said it would consider withholding royalty payments*, the DepBank and CCGbank annotations create a dependency between *said* and the following clause. However, in DepBank the relation is between *said* and *consider*, whereas in CCGbank the relation is between *said* and *would*. We fixed this problem by defining the head of *would consider* to be *consider* rather than *would*, by changing the annotation of all the relevant lexical categories in the CCG lexicon (mainly those creating aux relations).

There are more subject relations in CCGbank than DepBank. In the previous example, CCGbank has a subject relation between *it* and *consider*, and also *it* and *would*, whereas DepBank only has the relation between *it* and *consider*. In practice this means ignoring a number of the subject dependencies output by the CCG parser.

Another example where the dependencies differ is the treatment of relative pronouns. For example, in *Sen. Mitchell, who had proposed the streamlining*, the subject of *proposed* is *Mitchell* in CCGbank but *who* in DepBank. Again, we implemented this change by fixing the head annotation in the lexical categories which apply to relative pronouns.

4.2 Post processing of the GR output

To obtain some idea of whether the schemes were converging, we performed the following oracle experiment. We took the CCG derivations from CCGbank corresponding to the sentences in DepBank, and forced the parser to produce gold-standard derivations, outputting the newly created GRs. Treating the DepBank GRs as a gold-standard, and comparing these with the CCGbank GRs, gave precision and recall scores of only 76.23% and 79.56% respectively (using the RASP evaluation tool). Thus given the current mapping, the perfect CCGbank parser would achieve an F-score of only 77.86% when evaluated against DepBank.

On inspecting the output, it was clear that a number of general rules could be applied to bring the schemes closer together, which was implemented as a post-processing script. The first set of changes deals with coordination. One significant difference between DepBank and CCGbank is the treatment of coordinations as arguments. Consider the example *The president and chief executive officer said the loss stems from several factors.* For both DepBank and the transformed CCGbank there are two conj GRs arising

from the coordination: (conj and president) and (conj and officer). The difference arises in the subject of *said*: in DepBank the subject is *and*: (ncsubj said and _), whereas in CCGbank there are two subjects: (ncsubj said president _) and (ncsubj said officer _). We deal with this difference by replacing any pairs of GRs which differ only in their arguments, and where the arguments are coordinated items, with a single GR containing the coordination term as the argument.

Ampersands are a frequently occurring problem in WSJ text. For example, the CCGbank analysis of *Standard & Poor's index* assigns the lexical category N/N to both *Standard* and *&*, treating them as modifiers of *Poor*, whereas DepBank treats *&* as a coordinating term. We fixed this by creating conj GRs between any *&* and the two words either side; removing the modifier GR between the two words; and replacing any GRs in which the words either side of the *&* are arguments with a single GR in which *&* is the argument.

The ta relation, which identifies text adjuncts delimited by punctuation, is difficult to assign correctly to the parser output. The simple punctuation rules used by the parser do not contain enough information to distinguish between the various cases of ta. Thus the only rule we have implemented, which is somewhat specific to the newspaper genre, is to replace GRs of the form (cmod _ say arg) with (ta quote arg say), where say can be any of *say*, *said* or *says*. This rule applies to only a small subset of the ta cases but has high enough precision to be worthy of inclusion.

A common source of error is the distinction between iobj and ncmod, which is not surprising given the difficulty that human annotators have in distinguishing arguments and adjuncts. There are many cases where an argument in DepBank is an adjunct in CCGbank, and vice versa. The only change we have made is to turn all ncmod GRs with *of* as the modifier into iobj GRs (unless the ncmod is a partitive predeterminer). This was found to have high precision and applies to a large number of cases.

There are some dependencies in CCGbank which do not appear in DepBank. Examples include any dependencies in which a punctuation mark is one of the arguments; these were removed from the output.

We attempt to fill the subtype slot for some GRs. The subtype slot specifies additional information about the GR; examples include the value obj in a passive ncsubj, indicating that the subject is an underlying object; the value num in ncmod, indicating a numerical quantity; and prt in ncmod to indicate a verb particle. The passive case is identified as follows: any lexical category which starts $S[pss]\backslash NP$ indicates a passive verb, and we also mark any verbs POS tagged VBN and assigned the lexical category N/N as passive. Both these rules have high precision, but still leave many of the cases in DepBank unidentified. The numerical case is identified using two rules: the num subtype is added if any argument in a GR is assigned the lexical category $N/N[num]$, and if any of the arguments in an ncmod is POS tagged CD. prt is added to an ncmod if the modifiee has any of the verb POS tags and if the modifier has POS tag RP.

The final columns of Table 4 show the accuracy of the transformed gold-standard CCGbank dependencies when compared with DepBank; the simple post-processing rules have increased the F-score from 77.86% to 84.76%. This F-score is an *upper bound* on the performance of the CCG parser.

5 Results

The results in Table 4 were obtained by parsing the sentences from CCGbank corresponding to those in the 560-sentence test set used by Briscoe et al. (2006). We used the CCGbank sentences because these differ in some ways to the original Penn Treebank sentences (there are no quotation marks in CCGbank, for example) and the parser has been trained on CCGbank. Even here we experienced some unexpected difficulties, since some of the tokenisation is different between DepBank and CCGbank and there are some sentences in DepBank which have been significantly shortened compared to the original Penn Treebank sentences. We modified the CCGbank sentences — and the CCGbank analyses since these were used for the oracle experiments — to be as close to the DepBank sentences as possible. All the results were obtained using the RASP evaluation scripts, with the results for the RASP parser taken from Briscoe et al. (2006). The results for CCGbank were obtained using the oracle method described above.

	RASP			CCG parser			CCGbank			
Relation	Prec	Rec	F	Prec	Rec	F	Prec	Rec	F	# GRs
aux	93.33	91.00	**92.15**	94.20	89.25	91.66	96.47	90.33	93.30	400
conj	72.39	72.27	72.33	79.73	77.98	**78.84**	83.07	80.27	81.65	595
ta	42.61	51.37	**46.58**	52.31	11.64	19.05	62.07	12.59	20.93	292
det	87.73	90.48	89.09	95.25	95.42	**95.34**	97.27	94.09	95.66	1 114
ncmod	75.72	69.94	72.72	75.75	79.27	**77.47**	78.88	80.64	79.75	3 550
xmod	53.21	46.63	**49.70**	43.46	52.25	47.45	56.54	60.67	58.54	178
cmod	45.95	30.36	36.56	51.50	61.31	**55.98**	64.77	69.09	66.86	168
pmod	30.77	33.33	**32.00**	0.00	0.00	0.00	0.00	0.00	0.00	12
ncsubj	79.16	67.06	72.61	83.92	75.92	**79.72**	88.86	78.51	83.37	1 354
xsubj	33.33	28.57	**30.77**	0.00	0.00	0.00	50.00	28.57	36.36	7
csubj	12.50	50.00	**20.00**	0.00	0.00	0.00	0.00	0.00	0.00	2
dobj	83.63	79.08	81.29	87.03	89.40	**88.20**	92.11	90.32	91.21	1 764
obj2	23.08	30.00	26.09	65.00	65.00	**65.00**	66.67	60.00	63.16	20
iobj	70.77	76.10	73.34	77.60	70.04	**73.62**	83.59	69.81	76.08	544
xcomp	76.88	77.69	**77.28**	76.68	77.69	77.18	80.00	78.49	79.24	381
ccomp	46.44	69.42	55.55	79.55	72.16	**75.68**	80.81	76.31	78.49	291
pcomp	72.73	66.67	**69.57**	0.00	0.00	0.00	0.00	0.00	0.00	24
macroaverage	62.12	63.77	62.94	65.61	63.28	**64.43**	71.73	65.85	68.67	
microaverage	77.66	74.98	76.29	82.44	81.28	**81.86**	86.86	82.75	84.76	

Table 4: Accuracy on DepBank. F-score is the balanced harmonic mean of precision (P) and recall (R): $2PR/(P+R)$. # GRs is the number of GRs in DepBank.

The CCG parser results are based on automatically assigned POS tags, using the Curran and Clark (2003) tagger. The coverage of the parser on DepBank is 100%. For a GR in the parser output to be correct, it has to match the gold-standard GR exactly, including any subtype slots; however, it is possible for a GR to be incorrect at one level but correct at a *subsuming level*.[1] For example, if an ncmod GR is incorrectly labelled with xmod, but is otherwise correct, it will be correct for all levels which subsume both ncmod and xmod, for example mod. The micro-averaged scores are calculated by aggregating the counts for all the relations in the hierarchy, including the subsuming relations; the macro-averaged scores are the mean of the individual scores for each relation (Briscoe et al., 2006).

The results show that the performance of the CCG parser is higher than RASP overall, and also higher on the majority of GR types (especially the more frequent types). RASP uses an unlexicalised parsing model and has not been tuned to newspaper text. On the other hand it has had many years of development; thus it provides a strong baseline for this test set. The overall F-score for the CCG parser, 81.86%, is only 3 points below that for CCGbank, which provides an upper bound for the CCG parser (given the conversion process being used).

[1]The GRs are arranged in a hierarchy, with those in Table 1 at the leaves; a small number of more general GRs subsume these (Briscoe and Carroll, 2006).

6 Conclusion

A contribution of this paper has been to highlight the difficulties associated with cross-formalism parser comparison. Note that the difficulties are not unique to CCG, and many would apply to any cross-formalism comparison, especially with parsers using automatically extracted grammars. Parser evaluation has improved on the original Parseval measures (Carroll et al., 1998), but the challenge remains to develop a representation and evaluation suite which can be easily applied to a wide variety of parsers and formalisms. Despite the difficulties, we have given the first evaluation of a CCG parser outside of CCGbank, outperforming the RASP parser by over 5% overall and on the majority of dependency types.

Can the CCG parser be compared with parsers other than RASP? Briscoe and Carroll (2006) give a rough comparison of RASP with the Parc LFG parser on the different versions of DepBank, obtaining similar results overall, but they acknowledge that the results are not strictly comparable because of the different annotation schemes used. Comparison with Penn Treebank parsers would be difficult because, for many constructions, the Penn Treebank trees and

CCG derivations are different shapes, and reversing the mapping Hockenmaier used to create CCGbank would be very difficult. Hence we challenge other parser developers to map their own parse output into the version of DepBank used here.

One aspect of parser evaluation not covered in this paper is efficiency. The CCG parser took only 22.6 seconds to parse the 560 sentences in DepBank, with the accuracy given earlier. Using a cluster of 18 machines we have also parsed the entire Gigaword corpus in less than five days. Hence, we conclude that accurate, large-scale, linguistically-motivated NLP is now practical with CCG.

Acknowledgements

We would like to thanks the anonymous reviewers for their helpful comments. James Curran was funded under ARC Discovery grants DP0453131 and DP0665973.

References

Rens Bod. 2003. An efficient implementation of a new DOP model. In *Proceedings of the 10th Meeting of the EACL*, pages 19–26, Budapest, Hungary.

Ted Briscoe and John Carroll. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the PARC DepBank. In *Proceedings of the Poster Session of COLING/ACL-06*, Sydney, Australia.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the Interactive Demo Session of COLING/ACL-06*, Sydney, Australia.

Sabine Buchholz, Jorn Veenstra, and Walter Daelemans. 1999. Cascaded grammatical relation assignment. In *Proceedings of EMNLP/VLC-99*, pages 239–246, University of Maryland, June 21-22.

A. Cahill, M. Burke, R. O'Donovan, J. van Genabith, and A. Way. 2004. Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based LFG approximations. In *Proceedings of the 42nd Meeting of the ACL*, pages 320–327, Barcelona, Spain.

John Carroll, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proceedings of the 1st LREC Conference*, pages 447–454, Granada, Spain.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the ACL*, University of Michigan, Ann Arbor.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of the NAACL*, pages 132–139, Seattle, WA.

Stephen Clark and James R. Curran. 2004a. The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of COLING-04*, pages 282–288, Geneva, Switzerland.

Stephen Clark and James R. Curran. 2004b. Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Meeting of the ACL*, pages 104–111, Barcelona, Spain.

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

James R. Curran and Stephen Clark. 2003. Investigating GIS and smoothing for maximum entropy taggers. In *Proceedings of the 10th Meeting of the EACL*, pages 91–98, Budapest, Hungary.

Julia Hockenmaier and Mark Steedman. 2002. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Meeting of the ACL*, pages 335–342, Philadelphia, PA.

Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Ron Kaplan, Stefan Riezler, Tracy H. King, John T. Maxwell III, Alexander Vasserman, and Richard Crouch. 2004. Speed and accuracy in shallow and deep stochastic parsing. In *Proceedings of the HLT Conference and the 4th NAACL Meeting (HLT-NAACL'04)*, Boston, MA.

Tracy H. King, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The PARC 700 Dependency Bank. In *Proceedings of the LINC-03 Workshop*, Budapest, Hungary.

Robert Malouf and Gertjan van Noord. 2004. Wide coverage parsing with stochastic attribute value grammars. In *Proceedings of the IJCNLP-04 Workshop: Beyond shallow analyses - Formalisms and statistical modeling for deep analyses*, Hainan Island, China.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of COLING-2004*, pages 64–70, Geneva, Switzerland.

Judita Preiss. 2003. Using grammatical relations to compare parsers. In *Proceedings of the 10th Meeting of the EACL*, pages 291–298, Budapest, Hungary.

Anoop Sarkar and Aravind Joshi. 2003. Tree-adjoining grammars and its application to statistical parsing. In Rens Bod, Remko Scha, and Khalil Sima'an, editors, *Data-oriented parsing*. CSLI.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA.

Kristina Toutanova, Christopher Manning, Stuart Shieber, Dan Flickinger, and Stephan Oepen. 2002. Parse disambiguation for a rich HPSG grammar. In *Proceedings of the First Workshop on Treebanks and Linguistic Theories*, pages 253–263, Sozopol, Bulgaria.

Frustratingly Easy Domain Adaptation

Hal Daumé III
School of Computing
University of Utah
Salt Lake City, Utah 84112
me@hal3.name

Abstract

We describe an approach to domain adaptation that is appropriate exactly in the case when one has enough "target" data to do slightly better than just using only "source" data. Our approach is incredibly simple, easy to implement as a preprocessing step (10 lines of Perl!) and outperforms state-of-the-art approaches on a range of datasets. Moreover, it is trivially extended to a multi-domain adaptation problem, where one has data from a variety of different domains.

1 Introduction

The task of domain adaptation is to develop learning algorithms that can be easily ported from one domain to another—say, from newswire to biomedical documents. This problem is particularly interesting in NLP because we are often in the situation that we have a large collection of labeled data in one "source" domain (say, newswire) but truly desire a model that performs well in a second "target" domain. The approach we present in this paper is based on the idea of transforming the domain adaptation learning problem into a standard supervised learning problem to which any standard algorithm may be applied (eg., maxent, SVMs, etc.). Our transformation is incredibly simple: we augment the feature space of both the source and target data and use the result as input to a standard learning algorithm.

There are roughly two varieties of the domain adaptation problem that have been addressed in the literature: the fully supervised case and the semi-supervised case. The fully supervised case models the following scenario. We have access to a large, annotated corpus of data from a source domain. In addition, we spend a little money to annotate a small corpus in the target domain. We want to leverage both annotated datasets to obtain a model that performs well on the target domain. The semi-supervised case is similar, but instead of having a small annotated target corpus, we have a large but *unannotated* target corpus. In this paper, we focus exclusively on the fully supervised case.

One particularly nice property of our approach is that it is incredibly easy to implement: the Appendix provides a 10 line, 194 character Perl script for performing the complete transformation (available at `http://hal3.name/easyadapt.pl.gz`). In addition to this simplicity, our algorithm performs as well as (or, in some cases, better than) current state of the art techniques.

2 Problem Formalization and Prior Work

To facilitate discussion, we first introduce some notation. Denote by $\mathcal{X}$ the input space (typically either a real vector or a binary vector), and by $\mathcal{Y}$ the output space. We will write $\mathcal{D}^s$ to denote the distribution over source examples and $\mathcal{D}^t$ to denote the distribution over target examples. We assume access to a samples $D^s \sim \mathcal{D}^s$ of source examples from the source domain, and samples $D^t \sim \mathcal{D}^t$ of target examples from the target domain. We will assume that D^s is a collection of N examples and D^t is a collection of M examples (where, typically, $N \gg M$). Our goal is to learn a function $h : \mathcal{X} \to \mathcal{Y}$ with low expected loss with respect to the target domain.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 256–263,
Prague, Czech Republic, June 2007.

For the purposes of discussion, we will suppose that $\mathcal{X} = \mathbb{R}^F$ and that $\mathcal{Y} = \{-1, +1\}$. However, most of the techniques described in this section (as well as our own technique) are more general.

There are several "obvious" ways to attack the domain adaptation problem without developing new algorithms. Many of these are presented and evaluated by Daumé III and Marcu (2006).

The SRCONLY baseline ignores the target data and trains a single model, only on the source data.

The TGTONLY baseline trains a single model only on the target data.

The ALL baseline simply trains a standard learning algorithm on the union of the two datasets.

A potential problem with the ALL baseline is that if $N \gg M$, then D^s may "wash out" any affect D^t might have. We will discuss this problem in more detail later, but one potential solution is to re-weight examples from D^s. For instance, if $N = 10 \times M$, we may weight each example from the source domain by 0.1. The next baseline, WEIGHTED, is exactly this approach, with the weight chosen by cross-validation.

The PRED baseline is based on the idea of using the output of the source classifier as a feature in the target classifier. Specifically, we first train a SRCONLY model. Then we run the SRCONLY model on the target data (training, development and test). We use the predictions made by the SRCONLY model as additional features and train a second model on the target data, augmented with this new feature.

In the LININT baseline, we linearly interpolate the predictions of the SRCONLY and the TGTONLY models. The interpolation parameter is adjusted based on target development data.

These baselines are actually surprisingly difficult to beat. To date, there are two models that have successfully defeated them on a handful of datasets. The first model, which we shall refer to as the PRIOR model, was first introduced by Chelba and Acero (2004). The idea of this model is to use the SRCONLY model as a *prior* on the weights for a second model, trained on the target data. Chelba and Acero (2004) describe this approach within the context of a maximum entropy classifier, but the idea is more general. In particular, for many learning algorithms (maxent, SVMs, averaged perceptron, naive Bayes, etc.), one *regularizes* the weight vector toward zero. In other words, all of these algorithms contain a regularization term on the weights $\boldsymbol{w}$ of the form $\lambda ||\boldsymbol{w}||_2^2$. In the generalized PRIOR model, we simply replace this regularization term with $\lambda ||\boldsymbol{w} - \boldsymbol{w}^s||_2^2$, where $\boldsymbol{w}^s$ is the weight vector learned in the SRCONLY model.[1] In this way, the model trained on the target data "prefers" to have weights that are similar to the weights from the SRCONLY model, unless the data demands otherwise. Daumé III and Marcu (2006) provide empirical evidence on four datasets that the PRIOR model outperforms the baseline approaches.

More recently, Daumé III and Marcu (2006) presented an algorithm for domain adaptation for maximum entropy classifiers. The key idea of their approach is to learn *three* separate models. One model captures "source specific" information, one captures "target specific" information and one captures "general" information. The distinction between these three sorts of information is made on a *per-example* basis. In this way, each source example is considered either source specific or general, while each target example is considered either target specific or general. Daumé III and Marcu (2006) present an EM algorithm for training their model. This model consistently outperformed all the baseline approaches as well as the PRIOR model. Unfortunately, despite the empirical success of this algorithm, it is quite complex to implement and is roughly 10 to 15 times slower than training the PRIOR model.

3 Adaptation by Feature Augmentation

In this section, we describe our approach to the domain adaptation problem. Essentially, all we are going to do is take each feature in the original problem and make three versions of it: a general version, a source-specific version and a target-specific version. The augmented source data will contain only general and source-specific versions. The augmented target

[1] For the maximum entropy, SVM and naive Bayes learning algorithms, modifying the regularization term is simple because it appears explicitly. For the perceptron algorithm, one can obtain an equivalent regularization by performing standard perceptron updates, but using $(\boldsymbol{w} + \boldsymbol{w}^s)^\top \boldsymbol{x}$ for making predictions rather than simply $\boldsymbol{w}^\top \boldsymbol{x}$.

data contains general and target-specific versions.

To state this more formally, first recall the notation from Section 2: $\mathcal{X}$ and $\mathcal{Y}$ are the input and output spaces, respectively; D^s is the source domain data set and D^t is the target domain data set. Suppose for simplicity that $\mathcal{X} = \mathbb{R}^F$ for some $F > 0$. We will define our augmented input space by $\breve{\mathcal{X}} = \mathbb{R}^{3F}$. Then, define mappings $\Phi^s, \Phi^t : \mathcal{X} \to \breve{\mathcal{X}}$ for mapping the source and target data respectively. These are defined by Eq (1), where $\mathbf{0} = \langle 0, 0, \ldots, 0 \rangle \in \mathbb{R}^F$ is the zero vector.

$$\Phi^s(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{x}, \mathbf{0} \rangle, \quad \Phi^t(\boldsymbol{x}) = \langle \boldsymbol{x}, \mathbf{0}, \boldsymbol{x} \rangle \tag{1}$$

Before we proceed with a formal analysis of this transformation, let us consider why it might be expected to work. Suppose our task is part of speech tagging, our source domain is the Wall Street Journal and our target domain is a collection of reviews of computer hardware. Here, a word like "the" should be tagged as a determiner in both cases. However, a word like "monitor" is more likely to be a verb in the WSJ and more likely to be a noun in the hardware corpus. Consider a simple case where $\mathcal{X} = \mathbb{R}^2$, where x_1 indicates if the word is "the" and x_2 indicates if the word is "monitor." Then, in $\breve{\mathcal{X}}$, $\breve{x}_1$ and $\breve{x}_2$ will be "general" versions of the two indicator functions, $\breve{x}_3$ and $\breve{x}_4$ will be source-specific versions, and $\breve{x}_5$ and $\breve{x}_6$ will be target-specific versions.

Now, consider what a learning algorithm could do to capture the fact that the appropriate tag for "the" remains constant across the domains, and the tag for "monitor" changes. In this case, the model can set the "determiner" weight vector to something like $\langle 1, 0, 0, 0, 0, 0 \rangle$. This places high weight on the common version of "the" and indicates that "the" is most likely a determiner, regardless of the domain. On the other hand, the weight vector for "noun" might look something like $\langle 0, 0, 0, 0, 0, 1 \rangle$, indicating that the word "monitor" is a noun *only* in the target domain. Similar, the weight vector for "verb" might look like $\langle 0, 0, 0, 1, 0, 0 \rangle$, indicating the "monitor" is a verb *only* in the source domain.

Note that this expansion is actually redundant. We could equally well use $\Phi^s(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{x} \rangle$ and $\Phi^t(\boldsymbol{x}) = \langle \boldsymbol{x}, \mathbf{0} \rangle$. However, it turns out that it is easier to analyze the first case, so we will stick with that. Moreover, the first case has the nice property that it is straightforward to generalize it to the multi-domain adaptation problem: when there are more than two domains. In general, for K domains, the augmented feature space will consist of $K+1$ copies of the original feature space.

3.1 A Kernelized Version

It is straightforward to derive a kernelized version of the above approach. We do not exploit this property in our experiments—all are conducted with a simple linear kernel. However, by deriving the kernelized version, we gain some insight into the method. For this reason, we sketch the derivation here.

Suppose that the data points x are drawn from a reproducing kernel Hilbert space $\mathcal{X}$ with kernel $K : \mathcal{X} \times \mathcal{X} \to \mathcal{R}$, with K positive semi-definite. Then, K can be written as the dot product (in $\mathcal{X}$) of two (perhaps infinite-dimensional) vectors: $K(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{X}}$. Define Φ^s and Φ^t in terms of Φ, as:

$$\begin{aligned} \Phi^s(x) &= \langle \Phi(x), \Phi(x), \mathbf{0} \rangle \\ \Phi^t(x) &= \langle \Phi(x), \mathbf{0}, \Phi(x) \rangle \end{aligned} \tag{2}$$

Now, we can compute the kernel product between Φ^s and Φ^t in the expanded RKHS by making use of the original kernel K. We denote the expanded kernel by $\breve{K}(x, x')$. It is simplest to first describe $\breve{K}(x, x')$ when x and x' are from the same domain, then analyze the case when the domain differs. When the domain is the same, we get: $\breve{K}(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{X}} + \langle \Phi(x), \Phi(x') \rangle_{\mathcal{X}} = 2K(x, x')$. When they are from different domains, we get: $\breve{K}(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{X}} = K(x, x')$. Putting this together, we have:

$$\breve{K}(x, x') = \begin{cases} 2K(x, x') & \text{same domain} \\ K(x, x') & \text{diff. domain} \end{cases} \tag{3}$$

This is an intuitively pleasing result. What it says is that—considering the kernel as a measure of similarity—data points from the same domain are "by default" twice as similar as those from different domains. Loosely speaking, this means that data points from the target domain have twice as much influence as source points when making predictions about test target data.

3.2 Analysis

We first note an obvious property of the feature-augmentation approach. Namely, it does not make learning harder, in a minimum Bayes error sense. A more interesting statement would be that it makes learning *easier*, along the lines of the result of (Ben-David et al., 2006) — note, however, that their results are for the "semi-supervised" domain adaptation problem and so do not apply directly. As yet, we do not know a proper formalism in which to analyze the fully supervised case.

It turns out that the feature-augmentation method is remarkably similar to the PRIOR model[2]. Suppose we learn feature-augmented weights in a classifier regularized by an ℓ_2 norm (eg., SVMs, maximum entropy). We can denote by w_s the sum of the "source" and "general" components of the learned weight vector, and by w_t the sum of the "target" and "general" components, so that w_s and w_t are the predictive weights for each task. Then, the regularization condition on the entire weight vector is approximately $||w_g||^2 + ||w_s - w_g||^2 + ||w_t - w_g||^2$, with free parameter w_g which can be chosen to minimize this sum. This leads to a regularizer proportional to $||w_s - w_t||^2$, akin to the PRIOR model.

Given this similarity between the feature-augmentation method and the PRIOR model, one might wonder why we expect our approach to do better. Our belief is that this occurs because we optimize w_s and w_t *jointly*, not sequentially. First, this means that we do not need to cross-validate to estimate good hyperparameters for each task (though in our experiments, we do not use any hyperparameters). Second, and more importantly, this means that the single supervised learning algorithm that is run is allowed to regulate the trade-off between source/target and general weights. In the PRIOR model, we are forced to use the prior variance on in the target learning scenario to do this ourselves.

3.3 Multi-domain adaptation

Our formulation is agnostic to the number of "source" domains. In particular, it may be the case that the source data actually falls into a variety of more specific domains. This is simple to account for in our model. In the two-domain case, we expanded the feature space from $\mathbb{R}^F$ to $\mathbb{R}^{3F}$. For a K-domain problem, we simply expand the feature space to $\mathbb{R}^{(K+1)F}$ in the obvious way (the "$+1$" corresponds to the "general domain" while each of the other $1 \ldots K$ correspond to a single task).

[2]Thanks an anonymous reviewer for pointing this out!

4 Results

In this section we describe experimental results on a wide variety of domains. First we describe the tasks, then we present experimental results, and finally we look more closely at a few of the experiments.

4.1 Tasks

All tasks we consider are sequence labeling tasks (either named-entity recognition, shallow parsing or part-of-speech tagging) on the following datasets:

ACE-NER. We use data from the 2005 Automatic Content Extraction task, restricting ourselves to the named-entity recognition task. The 2005 ACE data comes from 5 domains: Broadcast News (bn), Broadcast Conversations (bc), Newswire (nw), Weblog (wl), Usenet (un) and Converstaional Telephone Speech (cts).

CoNLL-NE. Similar to ACE-NER, a named-entity recognition task. The difference is: we use the 2006 ACE data as the source domain and the CoNLL 2003 NER data as the target domain.

PubMed-POS. A part-of-speech tagging problem on PubMed abstracts introduced by Blitzer et al. (2006). There are two domains: the source domain is the WSJ portion of the Penn Treebank and the target domain is PubMed.

CNN-Recap. This is a recapitalization task introduced by Chelba and Acero (2004) and also used by Daumé III and Marcu (2006). The source domain is newswire and the target domain is the output of an ASR system.

Treebank-Chunk. This is a shallow parsing task based on the data from the Penn Treebank. This data comes from a variety of domains: the standard WSJ domain (we use the same data as for CoNLL 2000), the ATIS switchboard domain, and the Brown corpus (which is, itself, assembled from six subdomains).

Treebank-Brown. This is identical to the Treebank-Chunk task, except that we consider all of the Brown corpus to be a single domain.

Task	Dom	# Tr	# De	# Te	# Ft
ACE-NER	bn	52,998	6,625	6,626	80k
	bc	38,073	4,759	4,761	109k
	nw	44,364	5,546	5,547	113k
	wl	35,883	4,485	4,487	109k
	un	35,083	4,385	4,387	96k
	cts	39,677	4,960	4,961	54k
CoNLL-NER	src	256,145	-	-	368k
	tgt	29,791	5,258	8,806	88k
PubMed-POS	src	950,028	-	-	571k
	tgt	11,264	1,987	14,554	39k
CNN-Recap	src	2,000,000	-	-	368k
	tgt	39,684	7,003	8,075	88k
Treebank-Chunk	wsj	191,209	29,455	38,440	94k
	swbd3	45,282	5,596	41,840	55k
	br-cf	58,201	8,307	7,607	144k
	br-cg	67,429	9,444	6,897	149k
	br-ck	51,379	6,061	9,451	121k
	br-cl	47,382	5,101	5,880	95k
	br-cm	11,696	1,324	1,594	51k
	br-cn	56,057	6,751	7,847	115k
	br-cp	55,318	7,477	5,977	112k
	br-cr	16,742	2,522	2,712	65k

Table 1: Task statistics; columns are task, domain, size of the training, development and test sets, and the number of unique features in the training set.

In all cases (except for CNN-Recap), we use roughly the same feature set, which has become somewhat standardized: lexical information (words, stems, capitalization, prefixes and suffixes), membership on gazetteers, etc. For the CNN-Recap task, we use identical feature to those used by both Chelba and Acero (2004) and Daumé III and Marcu (2006): the current, previous and next word, and 1-3 letter prefixes and suffixes.

Statistics on the tasks and datasets are in Table 1.

In all cases, we use the SEARN algorithm for solving the sequence labeling problem (Daumé III et al., 2007) with an underlying averaged perceptron classifier; implementation due to (Daumé III, 2004). For structural features, we make a second-order Markov assumption and only place a bias feature on the transitions. For simplicity, we optimize and report only on label accuracy (but require that our outputs be parsimonious: we do not allow "I-NP" to follow "B-PP," for instance). We do this for three reasons. First, our focus in this work is on building better learning algorithms and introducing a more complicated measure only serves to mask these effects. Second, it is arguable that a measure like F_1 is inappropriate for chunking tasks (Manning, 2006). Third, we can easily compute statistical significance over accuracies using McNemar's test.

4.2 Experimental Results

The full—somewhat daunting—table of results is presented in Table 2. The first two columns specify the task and domain. For the tasks with only a single source and target, we simply report results on the target. For the multi-domain adaptation tasks, we report results for each setting of the target (where all other data-sets are used as different "source" domains). The next set of eight columns are the *error rates* for the task, using one of the different techniques ("AUGMENT" is our proposed technique). For each row, the error rate of the best performing technique is bolded (as are all techniques whose performance is not statistically significantly different at the 95% level). The "T<S" column is contains a "+" whenever TGTONLY outperforms SRCONLY (this will become important shortly). The final column indicates when AUGMENT comes in first.[3]

There are several trends to note in the results. Excluding for a moment the "br-*" domains on the Treebank-Chunk task, our technique always performs best. Still excluding "br-*", the clear second-place contestant is the PRIOR model, a finding consistent with prior research. When we repeat the Treebank-Chunk task, but lumping all of the "br-*" data together into a single "brown" domain, the story reverts to what we expected before: our algorithm performs best, followed by the PRIOR method.

Importantly, this simple story breaks down on the Treebank-Chunk task for the eight sections of the Brown corpus. For these, our AUGMENT technique performs rather poorly. Moreover, there is no clear winning approach on this task. Our hypothesis is that the common feature of these examples is that these are exactly the tasks for which SRCONLY outperforms TGTONLY (with one exception: CoNLL). This seems like a plausible explanation, since it implies that the source and target domains may not be that different. If the domains are so similar that a large amount of source data outperforms a small amount of target data, then it is unlikely that blow-

[3]One advantage of using the averaged perceptron for all experiments is that the only tunable hyperparameter is the number of iterations. In all cases, we run 20 iterations and choose the one with the lowest error on development data.

Task	Dom	SRCONLY	TGTONLY	ALL	WEIGHT	PRED	LININT	PRIOR	AUGMENT	T<S	Win
	bn	4.98	2.37	2.29	2.23	2.11	2.21	2.06	**1.98**	+	+
	bc	4.54	4.07	3.55	3.53	3.89	4.01	**3.47**	**3.47**	+	+
ACE-	nw	4.78	3.71	3.86	3.65	3.56	3.79	3.68	**3.39**	+	+
NER	wl	2.45	2.45	**2.12**	**2.12**	2.45	2.33	2.41	**2.12**	=	+
	un	3.67	2.46	2.48	2.40	2.18	2.10	2.03	**1.91**	+	+
	cts	2.08	0.46	0.40	0.40	0.46	0.44	**0.34**	**0.32**	+	+
CoNLL	tgt	2.49	2.95	1.80	**1.75**	2.13	**1.77**	1.89	**1.76**		+
PubMed	tgt	12.02	4.15	5.43	4.15	4.14	3.95	3.99	**3.61**	+	+
CNN	tgt	10.29	3.82	3.67	3.45	3.46	3.44	**3.35**	**3.37**	+	+
	wsj	6.63	4.35	4.33	4.30	4.32	4.32	4.27	**4.11**	+	+
	swbd3	15.90	4.15	4.50	4.10	4.13	4.09	3.60	**3.51**	+	+
	br-cf	5.16	6.27	4.85	4.80	4.78	**4.72**	5.22	5.15		
Tree	br-cg	4.32	5.36	**4.16**	**4.15**	4.27	4.30	4.25	4.90		
bank-	br-ck	5.05	6.32	5.05	4.98	**5.01**	**5.05**	5.27	5.41		
Chunk	br-cl	5.66	6.60	5.42	**5.39**	**5.39**	5.53	5.99	5.73		
	br-cm	3.57	6.59	**3.14**	**3.11**	3.15	3.31	4.08	4.89		
	br-cn	4.60	5.56	4.27	4.22	**4.20**	**4.19**	4.48	4.42		
	br-cp	4.82	5.62	4.63	**4.57**	**4.55**	**4.55**	4.87	4.78		
	br-cr	5.78	9.13	5.71	5.19	5.20	**5.15**	6.71	6.30		
Treebank-brown		6.35	5.75	4.80	4.75	4.81	4.72	4.72	**4.65**	+	+

Table 2: Task results.

ing up the feature space will help.

We additionally ran the MEGAM model (Daumé III and Marcu, 2006) on these data (though not in the multi-conditional case; for this, we considered the single source as the union of all sources). The results are not displayed in Table 2 to save space. For the majority of results, MEGAM performed roughly comparably to the best of the systems in the table. In particular, it was not statistically significantly different that AUGMENT on: ACE-NER, CoNLL, PubMed, Treebank-chunk-wsj, Treebank-chunk-swbd3, CNN and Treebank-brown. It did outperform AUGMENT on the Treebank-chunk on the Treebank-chunk-br-* data sets, but only outperformed the best other model on these data sets for br-cg, br-cm and br-cp. However, despite its advantages on these data sets, it was quite significantly slower to train: a single run required about ten times longer than any of the other models (including AUGMENT), and also required five-to-ten iterations of cross-validation to tune its hyperparameters so as to achieve these results.

4.3 Model Introspection

One explanation of our model's improved performance is simply that by augmenting the feature space, we are creating a more powerful model. While this may be a partial explanation, here we show that what the model learns about the various domains actually makes some plausible sense.

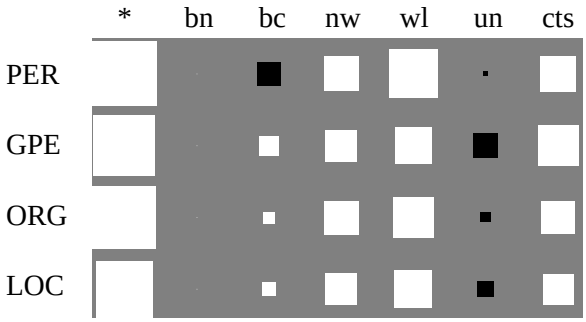


Figure 1: Hinton diagram for feature /Aa+/ at current position.

We perform this analysis only on the ACE-NER data by looking specifically at the learned weights. That is, for any given feature f, there will be seven versions of f: one corresponding to the "cross-domain" f and seven corresponding to each domain. We visualize these weights, using Hinton diagrams, to see how the weights vary across domains.

For example, consider the feature "current word has an initial capital letter and is then followed by one or more lower-case letters." This feature is presumably useless for data that lacks capitalization information, but potentially quite useful for other domains. In Figure 1 we shown a Hinton diagram for this figure. Each column in this figure correspond to a domain (the top row is the "general domain").

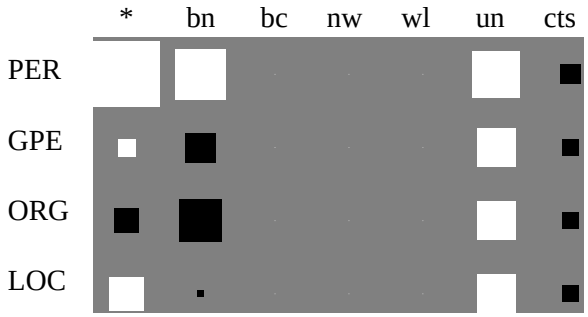


Figure 2: Hinton diagram for feature /bush/ at current position.

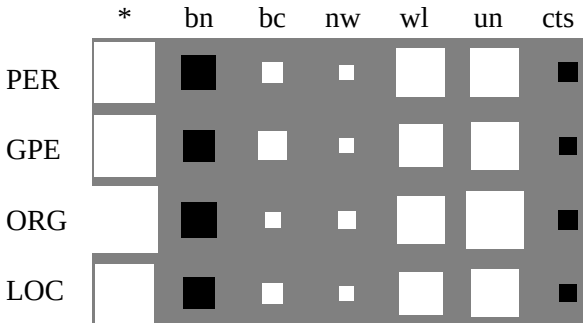


Figure 3: Hinton diagram for feature /the/ at current position.

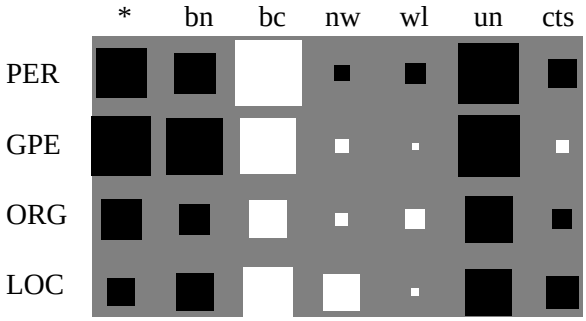


Figure 4: Hinton diagram for feature /the/ at previous position.

Each row corresponds to a class.[4] Black boxes correspond to negative weights and white boxes correspond to positive weights. The size of the box depicts the absolute value of the weight.

As we can see from Figure 1, the /Aa+/ feature is a very good indicator of entity-hood (it's value is strongly positive for all four entity classes), regardless of domain (i.e., for the "*" domain). The lack of boxes in the "bn" column means that, beyond the settings in "*", the broadcast news is agnostic with respect to this feature. This makes sense: there is no capitalization in broadcast news domain, so there would be no sense is setting these weights to anything by zero. The usenet column is filled with negative weights. While this may seem strange, it is due to the fact that many email addresses and URLs match this pattern, but are not entities.

Figure 2 depicts a similar figure for the feature "word is 'bush' at the current position" (this figure is case sensitive).[5] These weights are somewhat harder to interpret. What is happening is that "by default" the word "bush" is going to be a person—this is because it rarely appears referring to a plant and so even in the capitalized domains like broadcast conversations, if it appears at all, it is a person. The exception is that in the conversations data, people *do* actually talk about bushes as plants, and so the weights are set accordingly. The weights are high in the usenet domain because people tend to talk about the president without capitalizing his name.

Figure 3 presents the Hinton diagram for the feature "word at the current position is 'the"' (again, case-sensitive). In general, it appears, "the" is a common word in entities in all domain except for broadcast news and conversations. The exceptions are broadcast news and conversations. These exceptions crop up because of the capitalization issue.

In Figure 4, we show the diagram for the feature "previous word is 'the'." The only domain for which this is a good feature of entity-hood is broadcast conversations (to a much lesser extent, newswire). This occurs because of four phrases very common in the broadcast conversations and rare elsewhere: "the Iraqi people" ("Iraqi" is a GPE), "the Pentagon" (an ORG), "the Bush (cabinet|advisors|...)" (PER), and "the South" (LOC).

Finally, Figure 5 shows the Hinton diagram for the feature "the current word is on a list of common names" (this feature is case-*in*sensitive). All around, this is a good feature for picking out people and nothing else. The two exceptions are: it is also a good feature for other entity types for broadcast

[4]Technically there are many more classes than are shown here. We do not depict the smallest classes, and have merged the "Begin-*" and "In-*" weights for each entity type.

[5]The scale of weights across features is *not* comparable, so do not try to compare Figure 1 with Figure 2.

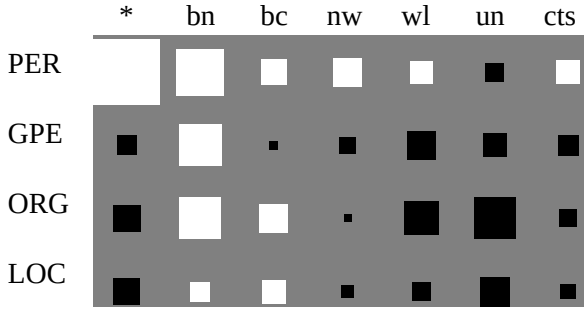


Figure 5: Hinton diagram for membership on a list of names at current position.

news and it is not quite so good for people in usenet. The first is easily explained: in broadcast news, it is very common to refer to countries and organizations by the name of their respective leaders. This is essentially a metonymy issue, but as the data is annotated, these are marked by their true referent. For usenet, it is because the list of names comes from news data, but usenet names are more diverse.

In general, the weights depicte for these features make some intuitive sense (in as much as weights for any learned algorithm make intuitive sense). It is particularly interesting to note that while there are some regularities to the patterns in the five diagrams, it is definitely *not* the case that there are, eg., two domains that behave identically across all features. This supports the hypothesis that the reason our algorithm works so well on this data is because the domains are actually quite well separated.

5 Discussion

In this paper we have described an *incredibly* simple approach to domain adaptation that—under a common and easy-to-verify condition—outperforms previous approaches. While it is somewhat frustrating that something so simple does so well, it is perhaps not surprising. By augmenting the feature space, we are essentially forcing the learning algorithm to do the adaptation for us. Good supervised learning algorithms have been developed over decades, and so we are essentially just leveraging all that previous work. Our hope is that this approach is so simple that it can be used for many more real-world tasks than we have presented here with little effort. Finally, it is very interesting to note that using our method, shallow parsing error rate on the CoNLL section of the treebank improves from 5.35 to 5.11. While this improvement is small, it is real, and may carry over to full parsing. The most important avenue of future work is to develop a formal framework under which we can analyze this (and other supervised domain adaptation models) theoretically. Currently our results only state that this augmentation procedure doesn't make the learning harder — we would like to know that it actually makes it easier. An additional future direction is to explore the kernelization interpretation further: why should we use 2 as the "similarity" between domains—we could introduce a hyperparamter α that indicates the similarity between domains and could be tuned via cross-validation.

Acknowledgments. We thank the three anonymous reviewers, as well as Ryan McDonald and John Blitzer for very helpful comments and insights.

References

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2006. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems (NIPS)*.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ciprian Chelba and Alex Acero. 2004. Adaptation of maximum entropy classifier: Little data can help a lot. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Barcelona, Spain.

Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26.

Hal Daumé III, John Langford, and Daniel Marcu. 2007. Search-based structured prediction. *Machine Learning Journal (submitted)*.

Hal Daumé III. 2004. Notes on CG and LM-BFGS optimization of logistic regression. Paper available at `http://pub.hal3.name/#daume04cg-bfgs`, implementation available at `http://hal3.name/megam/`, August.

Christopher Manning. 2006. Doing named entity recognition? Don't optimize for F_1. Post on the NLPers Blog, 25 August. `http://nlpers.blogspot.com/2006/08/doing-named-entity-recognition-dont.html`.

Instance Weighting for Domain Adaptation in NLP

Jing Jiang and **ChengXiang Zhai**
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{jiang4,czhai}@cs.uiuc.edu

Abstract

Domain adaptation is an important problem in natural language processing (NLP) due to the lack of labeled data in novel domains. In this paper, we study the domain adaptation problem from the instance weighting perspective. We formally analyze and characterize the domain adaptation problem from a distributional view, and show that there are two distinct needs for adaptation, corresponding to the different distributions of instances and classification functions in the source and the target domains. We then propose a general instance weighting framework for domain adaptation. Our empirical results on three NLP tasks show that incorporating and exploiting more information from the target domain through instance weighting is effective.

1 Introduction

Many natural language processing (NLP) problems such as part-of-speech (POS) tagging, named entity (NE) recognition, relation extraction, and semantic role labeling, are currently solved by supervised learning from manually labeled data. A bottleneck problem with this supervised learning approach is the lack of annotated data. As a special case, we often face the situation where we have a sufficient amount of labeled data in one domain, but have little or no labeled data in another related domain which we are interested in. We thus face the *domain adaptation* problem. Following (Blitzer et al., 2006), we call the first the *source domain*, and the second the *target domain*.

The domain adaptation problem is commonly encountered in NLP. For example, in POS tagging, the source domain may be tagged WSJ articles, and the target domain may be scientific literature that contains scientific terminology. In NE recognition, the source domain may be annotated news articles, and the target domain may be personal blogs. Another example is personalized spam filtering, where we may have many labeled spam and ham emails from publicly available sources, but we need to adapt the learned spam filter to an individual user's inbox because the user has her own, and presumably very different, distribution of emails and notion of spams.

Despite the importance of domain adaptation in NLP, currently there are no standard methods for solving this problem. An immediate possible solution is semi-supervised learning, where we simply treat the target instances as unlabeled data but do not distinguish the two domains. However, given that the source data and the target data are from different distributions, we should expect to do better by exploiting the domain difference. Recently there have been some studies addressing domain adaptation from different perspectives (Roark and Bacchiani, 2003; Chelba and Acero, 2004; Florian et al., 2004; Daumé III and Marcu, 2006; Blitzer et al., 2006). However, there have not been many studies that focus on the difference between the instance distributions in the two domains. A detailed discussion on related work is given in Section 5.

In this paper, we study the domain adaptation problem from the instance weighting perspective.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 264–271,
Prague, Czech Republic, June 2007.

In general, the domain adaptation problem arises when the source instances and the target instances are from two different, but related distributions. We formally analyze and characterize the domain adaptation problem from this distributional view. Such an analysis reveals that there are two distinct needs for adaptation, corresponding to the different distributions of instances and the different classification functions in the source and the target domains. Based on this analysis, we propose a general instance weighting method for domain adaptation, which can be regarded as a generalization of an existing approach to semi-supervised learning. The proposed method implements several adaptation heuristics with a unified objective function: (1) removing misleading training instances in the source domain; (2) assigning more weights to labeled target instances than labeled source instances; (3) augmenting training instances with target instances with predicted labels. We evaluated the proposed method with three adaptation problems in NLP, including POS tagging, NE type classification, and spam filtering. The results show that regular semi-supervised and supervised learning methods do not perform as well as our new method, which explicitly captures domain difference. Our results also show that incorporating and exploiting more information from the target domain is much more useful for improving performance than excluding misleading training examples from the source domain.

The rest of the paper is organized as follows. In Section 2, we formally analyze the domain adaptation problem and distinguish two types of adaptation. In Section 3, we then propose a general instance weighting framework for domain adaptation. In Section 4, we present the experiment results. Finally, we compare our framework with related work in Section 5 before we conclude in Section 6.

2 Domain Adaptation

In this section, we define and analyze domain adaptation from a theoretical point of view. We show that the need for domain adaptation arises from two factors, and the solutions are different for each factor. We restrict our attention to those NLP tasks that can be cast into multiclass classification problems, and we only consider discriminative models for classification. Since both are common practice in NLP, our analysis is applicable to many NLP tasks.

Let $\mathcal{X}$ be a feature space we choose to represent the observed instances, and let $\mathcal{Y}$ be the set of class labels. In the standard supervised learning setting, we are given a set of labeled instances $\{(x_i, y_i)\}_{i=1}^N$, where $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$, and (x_i, y_i) are drawn from an unknown joint distribution $p(x, y)$. Our goal is to recover this unknown distribution so that we can predict unlabeled instances drawn from the same distribution. In discriminative models, we are only concerned with $p(y|x)$. Following the maximum likelihood estimation framework, we start with a parameterized model family $p(y|x;\theta)$, and then find the best model parameter θ^* that maximizes the expected log likelihood of the data:

$$\theta^* = \arg\max_\theta \int_{\mathcal{X}} \sum_{y\in\mathcal{Y}} p(x,y) \log p(y|x;\theta) dx.$$

Since we do not know the distribution $p(x, y)$, we maximize the empirical log likelihood instead:

$$\begin{aligned}\theta^* &\approx \arg\max_\theta \int_{\mathcal{X}} \sum_{y\in\mathcal{Y}} \tilde{p}(x,y) \log p(y|x;\theta) dx \\ &= \arg\max_\theta \frac{1}{N} \sum_{i=1}^{N} \log p(y_i|x_i;\theta).\end{aligned}$$

Note that since we use the empirical distribution $\tilde{p}(x, y)$ to approximate $p(x, y)$, the estimated θ^* is dependent on $\tilde{p}(x, y)$. In general, as long as we have sufficient labeled data, this approximation is fine because the unlabeled instances we want to classify are from the same $p(x, y)$.

2.1 Two Factors for Domain Adaptation

Let us now turn to the case of domain adaptation where the unlabeled instances we want to classify are from a different distribution than the labeled instances. Let $p_s(x, y)$ and $p_t(x, y)$ be the true underlying distributions for the source and the target domains, respectively. Our general idea is to use $p_s(x, y)$ to approximate $p_t(x, y)$ so that we can exploit the labeled examples in the source domain.

If we factor $p(x, y)$ into $p(x, y) = p(y|x)p(x)$, we can see that $p_t(x, y)$ can deviate from $p_s(x, y)$ in two different ways, corresponding to two different kinds of domain adaptation:

Case 1 (Labeling Adaptation): $p_t(y|x)$ deviates from $p_s(y|x)$ to a certain extent. In this case, it is clear that our estimation of $p_s(y|x)$ from the labeled source domain instances will not be a good estimation of $p_t(y|x)$, and therefore domain adaptation is needed. We refer to this kind of adaptation as *function/labeling adaptation*.

Case 2 (Instance Adaptation): $p_t(y|x)$ is mostly similar to $p_s(y|x)$, but $p_t(x)$ deviates from $p_s(x)$. In this case, it may appear that our estimated $p_s(y|x)$ can still be used in the target domain. However, as we have pointed out, the estimation of $p_s(y|x)$ depends on the empirical distribution $\tilde{p}_s(x, y)$, which deviates from $p_t(x, y)$ due to the deviation of $p_s(x)$ from $p_t(x)$. In general, the estimation of $p_s(y|x)$ would be more influenced by the instances with high $\tilde{p}_s(x, y)$ (i.e., high $\tilde{p}_s(x)$). If $p_t(x)$ is very different from $p_s(x)$, then we should expect $p_t(x, y)$ to be very different from $p_s(x, y)$, and therefore different from $\tilde{p}_s(x, y)$. We thus cannot expect the estimated $p_s(y|x)$ to work well on the regions where $p_t(x, y)$ is high, but $p_s(x, y)$ is low. Therefore, in this case, we still need domain adaptation, which we refer to as *instance adaptation*.

Because the need for domain adaptation arises from two different factors, we need different solutions for each factor.

2.2 Solutions for Labeling Adaptation

If $p_t(y|x)$ deviates from $p_s(y|x)$ to some extent, we have one of the following choices:

Change of representation:

It may be the case that if we change the representation of the instances, i.e., if we choose a feature space $\mathcal{X}'$ different from $\mathcal{X}$, we can bridge the gap between the two distributions $p_s(y|x)$ and $p_t(y|x)$. For example, consider domain adaptive NE recognition where the source domain contains clean newswire data, while the target domain contains broadcast news data that has been transcribed by automatic speech recognition and lacks capitalization. Suppose we use a naive NE tagger that only looks at the word itself. If we consider capitalization, then the instance *Bush* is represented differently from the instance *bush*. In the source domain, $p_s(y = \text{Person}|x = \textit{Bush})$ is high while $p_s(y = \text{Person}|x = \textit{bush})$ is low, but in the target domain, $p_t(y = \text{Person}|x = \textit{bush})$ is high. If we ignore the capitalization information, then in both domains $p(y = \text{Person}|x = \textit{bush})$ will be high provided that the source domain contains much fewer instances of *bush* than *Bush*.

Adaptation through prior:

When we use a parameterized model $p(y|x; \theta)$ to approximate $p(y|x)$ and estimate θ based on the source domain data, we can place some prior on the model parameter θ so that the estimated distribution $p(y|x; \hat{\theta})$ will be closer to $p_t(y|x)$. Consider again the NE tagging example. If we use capitalization as a feature, in the source domain where capitalization information is available, this feature will be given a large weight in the learned model because it is very useful. If we place a prior on the weight for this feature so that a large weight will be penalized, then we can prevent the learned model from relying too much on this domain specific feature.

Instance pruning:

If we know the instances x for which $p_t(y|x)$ is different from $p_s(y|x)$, we can actively remove these instances from the training data because they are "misleading".

For all the three solutions given above, we need either some prior knowledge about the target domain, or some labeled target domain instances; from only the unlabeled target domain instances, we would not know where and why $p_t(y|x)$ differs from $p_s(y|x)$.

2.3 Solutions for Instance Adaptation

In the case where $p_t(y|x)$ is similar to $p_s(y|x)$, but $p_t(x)$ deviates from $p_s(x)$, we may use the (unlabeled) target domain instances to bias the estimate of $p_s(x)$ toward a better approximation of $p_t(x)$, and thus achieve domain adaptation. We explain the idea below.

Our goal is to obtain a good estimate of θ_t^* that is optimized according to the *target* domain distribution $p_t(x, y)$. The exact objective function is thus

$$\begin{aligned}\theta_t^* &= \arg\max_\theta \int_{\mathcal{X}} \sum_{y\in\mathcal{Y}} p_t(x, y) \log p(y|x; \theta) dx \\ &= \arg\max_\theta \int_{\mathcal{X}} p_t(x) \sum_{y\in\mathcal{Y}} p_t(y|x) \log p(y|x; \theta) dx.\end{aligned}$$

Our idea of domain adaptation is to exploit the labeled instances in the *source* domain to help obtain θ_t^*.

Let $\mathcal{D}_s = \{(x_i^s, y_i^s)\}_{i=1}^{N_s}$ denote the set of labeled instances we have from the source domain. Assume that we have a (small) set of labeled and a (large) set of unlabeled instances from the target domain, denoted by $\mathcal{D}_{t,l} = \{(x_j^{t,l}, y_j^{t,l})\}_{j=1}^{N_{t,l}}$ and $\mathcal{D}_{t,u} = \{x_k^{t,u}\}_{k=1}^{N_{t,u}}$, respectively. We now show three ways to approximate the objective function above, corresponding to using three different sets of instances to approximate the instance space $\mathcal{X}$.

Using $\mathcal{D}_s$:

Using $p_s(y|x)$ to approximate $p_t(y|x)$, we obtain

$$\begin{aligned}\theta_t^* &\approx \arg\max_\theta \int_{\mathcal{X}} \frac{p_t(x)}{p_s(x)} p_s(x) \sum_{y\in\mathcal{Y}} p_s(y|x) \log p(y|x;\theta) dx \\ &\approx \arg\max_\theta \int_{\mathcal{X}} \frac{p_t(x)}{p_s(x)} \tilde{p}_s(x) \sum_{y\in\mathcal{Y}} \tilde{p}_s(y|x) \log p(y|x;\theta) dx \\ &= \arg\max_\theta \frac{1}{N_s} \sum_{i=1}^{N_s} \frac{p_t(x_i^s)}{p_s(x_i^s)} \log p(y_i^s|x_i^s;\theta).\end{aligned}$$

Here we use only the labeled instances in $\mathcal{D}_s$ but we adjust the weight of each instance by $\frac{p_t(x)}{p_s(x)}$. The major difficulty is how to accurately estimate $\frac{p_t(x)}{p_s(x)}$.

Using $\mathcal{D}_{t,l}$:

$$\begin{aligned}\theta_t^* &\approx \arg\max_\theta \int_{\mathcal{X}} \tilde{p}_{t,l}(x) \sum_{y\in\mathcal{Y}} \tilde{p}_{t,l}(y|x) \log p(y|x;\theta) dx \\ &= \arg\max_\theta \frac{1}{N_{t,l}} \sum_{j=1}^{N_{t,l}} \log p(y_j^{t,l}|x_j^{t,l};\theta)\end{aligned}$$

Note that this is the standard supervised learning method using only the small amount of labeled target instances. The major weakness of this approximation is that when $N_{t,l}$ is very small, the estimation is not accurate.

Using $\mathcal{D}_{t,u}$:

$$\begin{aligned}\theta_t^* &\approx \arg\max_\theta \int_{\mathcal{X}} \tilde{p}_{t,u}(x) \sum_{y\in\mathcal{Y}} p_t(y|x) \log p(y|x;\theta) dx \\ &= \arg\max_\theta \frac{1}{N_{t,u}} \sum_{k=1}^{N_{t,u}} \sum_{y\in\mathcal{Y}} p_t(y|x_k^{t,u}) \log p(y|x_k^{t,u};\theta),\end{aligned}$$

The challenge here is that $p_t(y|x_k^{t,u};\theta)$ is unknown to us, thus we need to estimate it. One possibility is to approximate it with a model $\hat{\theta}$ learned from $\mathcal{D}_s$ and $\mathcal{D}_{t,l}$. For example, we can set $p_t(y|x,\theta) = p(y|x;\hat{\theta})$. Alternatively, we can also set $p_t(y|x,\theta)$ to 1 if $y = \arg\max_{y'} p(y'|x;\hat{\theta})$ and 0 otherwise.

3 A Framework of Instance Weighting for Domain Adaptation

The theoretical analysis we give in Section 2 suggests that one way to solve the domain adaptation problem is through instance weighting. We propose a framework that incorporates instance pruning in Section 2.2 and the three approximations in Section 2.3. Before we show the formal framework, we first introduce some weighting parameters and explain the intuitions behind these parameters.

First, for each $(x_i^s, y_i^s) \in \mathcal{D}_s$, we introduce a parameter α_i to indicate how likely $p_t(y_i^s|x_i^s)$ is close to $p_s(y_i^s|x_i^s)$. Large α_i means the two probabilities are close, and therefore we can trust the labeled instance (x_i^s, y_i^s) for the purpose of learning a classifier for the target domain. Small α_i means these two probabilities are very different, and therefore we should probably discard the instance (x_i^s, y_i^s) in the learning process.

Second, again for each $(x_i^s, y_i^s) \in \mathcal{D}_s$, we introduce another parameter β_i that ideally is equal to $\frac{p_t(x_i^s)}{p_s(x_i^s)}$. From the approximation in Section 2.3 that uses only $\mathcal{D}_s$, it is clear that such a parameter is useful.

Next, for each $x_i^{t,u} \in \mathcal{D}_{t,u}$, and for each possible label $y \in \mathcal{Y}$, we introduce a parameter $\gamma_i(y)$ that indicates how likely we would like to assign y as a tentative label to $x_i^{t,u}$ and include $(x_i^{t,u}, y)$ as a training example.

Finally, we introduce three global parameters λ_s, $\lambda_{t,l}$ and $\lambda_{t,u}$ that are not instance-specific but are associated with $\mathcal{D}_s$, $\mathcal{D}_{t,l}$ and $\mathcal{D}_{t,u}$, respectively. These three parameters allow us to control the contribution of each of the three approximation methods in Section 2.3 when we linearly combine them together.

We now formally define our instance weighting framework. Given $\mathcal{D}_s$, $\mathcal{D}_{t,l}$ and $\mathcal{D}_{t,u}$, to learn a classifier for the target domain, we find a parameter $\hat{\theta}$ that optimizes the following objective function:

$$\hat{\theta} = \arg\max_{\theta} \Big[\lambda_s \cdot \frac{1}{C_s} \sum_{i=1}^{N_s} \alpha_i \beta_i \log p(y_i^s | x_i^s; \theta) + \lambda_{t,l} \cdot \frac{1}{C_{t,l}} \sum_{j=1}^{N_{t,l}} \log p(y_j^{t,l} | x_j^{t,l}; \theta) + \lambda_{t,u} \cdot \frac{1}{C_{t,u}} \sum_{k=1}^{N_{t,u}} \sum_{y \in \mathcal{Y}} \gamma_k(y) \log p(y | x_k^{t,u}; \theta) + \log p(\theta) \Big],$$

where $C_s = \sum_{i=1}^{N_s} \alpha_i \beta_i$, $C_{t,l} = N_{t,l}$, $C_{t,u} = \sum_{k=1}^{N_{t,u}} \sum_{y \in \mathcal{Y}} \gamma_k(y)$, and $\lambda_s + \lambda_{t,l} + \lambda_{t,u} = 1$. The last term, $\log p(\theta)$, is the log of a Gaussian prior distribution of θ, commonly used to regularize the complexity of the model.

In general, we do not know the optimal values of these parameters for the target domain. Nevertheless, the intuitions behind these parameters serve as guidelines for us to design heuristics to set these parameters. In the rest of this section, we introduce several heuristics that we used in our experiments to set these parameters.

3.1 Setting α

Following the intuition that if $p_t(y|x)$ differs much from $p_s(y|x)$, then (x, y) should be discarded from the training set, we use the following heuristic to set α^s. First, with standard supervised learning, we train a model $\hat{\theta}_{t,l}$ from $\mathcal{D}_{t,l}$. We consider $p(y|x; \hat{\theta}_{t,l})$ to be a crude approximation of $p_t(y|x)$. Then, we classify $\{x_i^s\}_{i=1}^{N_s}$ using $\hat{\theta}_{t,l}$. The top k instances that are incorrectly predicted by $\hat{\theta}_{t,l}$ (ranked by their prediction confidence) are discarded. In another word, α_i^s of the top k instances for which $y_i^s \neq \arg\max_y p(y|x_i^s; \hat{\theta}_{t,l})$ are set to 0, and α_i of all the other source instances are set to 1.

3.2 Setting β

Accurately setting β involves accurately estimating $p_s(x)$ and $p_t(x)$ from the empirical distributions. For many NLP classification tasks, we do not have a good parametric model for $p(x)$. We thus need to resort to non-parametric density estimation methods. However, for many NLP tasks, x resides in a high dimensional space, which makes it hard to apply standard non-parametric density estimation methods. We have not explored this direction, and in our experiments, we set β to 1 for all source instances.

3.3 Setting γ

Setting γ is closely related to some semi-supervised learning methods. One option is to set $\gamma_k(y) = p(y|x_k^{t,u}; \theta)$. In this case, γ is no longer a constant but is a function of θ. This way of setting γ corresponds to the entropy minimization semi-supervised learning method (Grandvalet and Bengio, 2005). Another way to set γ corresponds to bootstrapping semi-supervised learning. First, let $\hat{\theta}^{(n)}$ be a model learned from the previous round of training. We then select the top k instances from $\mathcal{D}_{t,u}$ that have the highest prediction confidence. For these instances, we set $\gamma_k(y) = 1$ for $y = \arg\max_{y'} p(y'|x_k^{t,u}; \hat{\theta}^{(n)})$, and $\gamma_k(y) = 0$ for all other y. In another word, we select the top k confidently predicted instances, and include these instances together with their predicted labels in the training set. All other instances in $\mathcal{D}_{t,u}$ are not considered. In our experiments, we only considered this bootstrapping way of setting γ.

3.4 Setting λ

λ_s, $\lambda_{t,l}$ and $\lambda_{t,u}$ control the balance among the three sets of instances. Using standard supervised learning, λ_s and $\lambda_{t,l}$ are set proportionally to C_s and $C_{t,l}$, that is, each instance is weighted the same whether it is in $\mathcal{D}_s$ or in $\mathcal{D}_{t,l}$, and $\lambda_{t,u}$ is set to 0. Similarly, using standard bootstrapping, $\lambda_{t,u}$ is set proportionally to $C_{t,u}$, that is, each target instance added to the training set is also weighted the same as a source instance. In neither case are the target instances emphasize more than source instances. However, for domain adaptation, we want to focus more on the target domain instances. So intuitively, we want to make $\lambda_{t,l}$ and $\lambda_{t,u}$ somehow larger relative to λ_s. As we will show in Section 4, this is indeed beneficial.

In general, the framework provides great flexibility for implementing different adaptation strategies through these instance weighting parameters.

4 Experiments

4.1 Tasks and Data Sets

We chose three different NLP tasks to evaluate our instance weighting method for domain adaptation. The first task is POS tagging, for which we used

6166 WSJ sentences from Sections 00 and 01 of Penn Treebank as the source domain data, and 2730 PubMed sentences from the Oncology section of the PennBioIE corpus as the target domain data. The second task is entity type classification. The setup is very similar to Daumé III and Marcu (2006). We assume that the entity boundaries have been correctly identified, and we want to classify the types of the entities. We used ACE 2005 training data for this task. For the source domain, we used the newswire collection, which contains 11256 examples, and for the target domains, we used the weblog (WL) collection (5164 examples) and the conversational telephone speech (CTS) collection (4868 examples). The third task is personalized spam filtering. We used the ECML/PKDD 2006 discovery challenge data set. The source domain contains 4000 spam and ham emails from publicly available sources, and the target domains are three individual users' inboxes, each containing 2500 emails.

For each task, we consider two experiment settings. In the first setting, we assume there are a small number of labeled target instances available. For POS tagging, we used an additional 300 Oncology sentences as labeled target instances. For NE typing, we used 500 labeled target instances and 2000 unlabeled target instances for each target domain. For spam filtering, we used 200 labeled target instances and 1800 unlabeled target instances. In the second setting, we assume there is no labeled target instance. We thus used all available target instances for testing in all three tasks.

We used logistic regression as our model of $p(y|x;\theta)$ because it is a robust learning algorithm and widely used.

We now describe three sets of experiments, corresponding to three heuristic ways of setting α, $\lambda_{t,l}$ and $\lambda_{t,u}$.

4.2 Removing "Misleading" Source Domain Instances

In the first set of experiments, we gradually remove "misleading" labeled instances from the source domain, using the small number of labeled target instances we have. We follow the heuristic we described in Section 3.1, which sets the α for the top k misclassified source instances to 0, and the α for all the other source instances to 1. We also set $\lambda_{t,l}$ and $\lambda_{t,l}$ to 0 in order to focus only on the effect of removing "misleading" instances. We compare with a baseline method which uses all source instances with equal weight but no target instances. The results are shown in Table 1.

From the table, we can see that in most experiments, removing these predicted "misleading" examples improved the performance over the baseline. In some experiments (Oncology, CTS, u00, u01), the largest improvement was achieved when all misclassified source instances were removed. In the case of weblog NE type classification, however, removing the source instances hurt the performance. A possible reason for this is that the set of labeled target instances we use is a biased sample from the target domain, and therefore the model trained on these instances is not always a good predictor of "misleading" source instances.

4.3 Adding Labeled Target Domain Instances with Higher Weights

The second set of experiments is to add the labeled target domain instances into the training set. This corresponds to setting $\lambda_{t,l}$ to some non-zero value, but still keeping $\lambda_{t,u}$ as 0. If we ignore the domain difference, then each labeled target instance is weighted the same as a labeled source instance ($\frac{\lambda_{u,l}}{\lambda_s} = \frac{C_{u,l}}{C_s}$), which is what happens in regular supervised learning. However, based on our theoretical analysis, we can expect the labeled target instances to be more representative of the target domain than the source instances. We can therefore assign higher weights for the target instances, by adjusting the ratio between $\lambda_{t,l}$ and λ_s. In our experiments, we set $\frac{\lambda_{t,l}}{\lambda_s} = a\frac{C_{t,l}}{C_s}$, where a ranges from 2 to 20. The results are shown in Table 2.

As shown from the table, adding some labeled target instances can greatly improve the performance for all tasks. And in almost all cases, weighting the target instances more than the source instances performed better than weighting them equally.

We also tested another setting where we first removed the "misleading" source examples as we showed in Section 4.2, and then added the labeled target instances. The results are shown in the last row of Table 2. However, although both removing "misleading" source instances and adding labeled

POS		NE Type				Spam			
k	**Oncology**	k	**CTS**	k	**WL**	k	**u00**	**u01**	**u02**
0	0.8630	0	0.7815	0	0.7045	0	0.6306	0.6950	0.7644
4000	0.8675	800	0.8245	600	0.7070	150	0.6417	0.7078	0.7950
8000	0.8709	1600	0.8640	1200	0.6975	300	0.6611	0.7228	0.8222
12000	0.8713	2400	0.8825	1800	0.6830	450	0.7106	0.7806	0.8239
16000	0.8714	3000	0.8825	2400	0.6795	600	0.7911	0.8322	0.8328
all	0.8720	all	0.8830	all	0.6600	all	0.8106	0.8517	0.8067

Table 1: Accuracy on the target domain after removing "misleading" source domain instances.

POS		NE Type			Spam			
method	**Oncology**	**method**	**CTS**	**WL**	**method**	**u00**	**u01**	**u02**
$\mathcal{D}_s$ only	0.8630	$\mathcal{D}_s$ only	0.7815	0.7045	$\mathcal{D}_s$ only	0.6306	0.6950	0.7644
$\mathcal{D}_s + \mathcal{D}_{t,l}$	0.9349	$\mathcal{D}_s + \mathcal{D}_{t,l}$	0.9340	0.7735	$\mathcal{D}_s + \mathcal{D}_{t,l}$	0.9572	0.9572	0.9461
$\mathcal{D}_s + 5\mathcal{D}_{t,l}$	0.9411	$\mathcal{D}_s + 2\mathcal{D}_{t,l}$	0.9355	0.7810	$\mathcal{D}_s + 2\mathcal{D}_{t,l}$	0.9606	0.9600	0.9533
$\mathcal{D}_s + 10\mathcal{D}_{t,l}$	0.9429	$\mathcal{D}_s + 5\mathcal{D}_{t,l}$	0.9360	0.7820	$\mathcal{D}_s + 5D_{t,l}$	0.9628	09611	0.9601
$\mathcal{D}_s + 20\mathcal{D}_{t,l}$	0.9443	$\mathcal{D}_s + 10\mathcal{D}_{t,l}$	0.9355	0.7840	$\mathcal{D}_s + 10\mathcal{D}_{t,l}$	0.9639	0.9628	0.9633
$\mathcal{D}'_s + 20\mathcal{D}_{t,l}$	0.9422	$\mathcal{D}'_s + 10\mathcal{D}_{t,l}$	0.8950	0.6670	$\mathcal{D}'_s + 10\mathcal{D}_{t,l}$	0.9717	0.9478	0.9494

Table 2: Accuracy on the unlabeled target instances after adding the labeled target instances.

target instances work well individually, when combined, the performance in most cases is not as good as when no source instances are removed. We hypothesize that this is because after we added some labeled target instances with large weights, we already gained a good balance between the source data and the target data. Further removing source instances would push the emphasis more on the set of labeled target instances, which is only a biased sample of the whole target domain.

The POS data set and the CTS data set have previously been used for testing other adaptation methods (Daumé III and Marcu, 2006; Blitzer et al., 2006), though the setup there is different from ours. Our performance using instance weighting is comparable to their best performance (slightly worse for POS and better for CTS).

4.4 Bootstrapping with Higher Weights

In the third set of experiments, we assume that we do not have any labeled target instances. We tried two bootstrapping methods. The first is a *standard* bootstrapping method, in which we gradually added the most confidently predicted unlabeled target instances with their predicted labels to the training set. Since we believe that the target instances should in general be given more weight because they better represent the target domain than the source instances, in the second method, we gave the added target instances more weight in the objective function. In particular, we set $\lambda_{t,u} = \lambda_s$ such that the total contribution of the added target instances is equal to that of all the labeled source instances. We call this second method the *balanced* bootstrapping method. Table 3 shows the results.

As we can see, while bootstrapping can generally improve the performance over the baseline where no unlabeled data is used, the balanced bootstrapping method performed slightly better than the standard bootstrapping method. This again shows that weighting the target instances more is a right direction to go for domain adaptation.

5 Related Work

There have been several studies in NLP that address domain adaptation, and most of them need labeled data from both the source domain and the target domain. Here we highlight a few representative ones.

For generative syntactic parsing, Roark and Bacchiani (2003) have used the source domain data to construct a Dirichlet prior for MAP estimation of the PCFG for the target domain. Chelba and Acero (2004) use the parameters of the maximum entropy model learned from the source domain as the means of a Gaussian prior when training a new model on the target data. Florian et al. (2004) first train a NE tagger on the source domain, and then use the tagger's predictions as features for training and testing on the target domain.

The only work we are aware of that directly mod-

	POS	NE Type		Spam		
method	**Oncology**	**CTS**	**WL**	**u00**	**u01**	**u02**
supervised	0.8630	0.7781	0.7351	0.6476	0.6976	0.8068
standard bootstrap	0.8728	0.8917	0.7498	0.8720	0.9212	0.9760
balanced bootstrap	0.8750	0.8923	0.7523	0.8816	0.9256	0.9772

Table 3: Accuracy on the target domain without using labeled target instances. In balanced bootstrapping, more weights are put on the target instances in the objective function than in standard bootstrapping.

els the different distributions in the source and the target domains is by Daumé III and Marcu (2006). They assume a "truly source domain" distribution, a "truly target domain" distribution, and a "general domain" distribution. The source (target) domain data is generated from a mixture of the "truly source (target) domain" distribution and the "general domain" distribution. In contrast, we do not assume such a mixture model.

None of the above methods would work if there were no labeled target instances. Indeed, all the above methods do not make use of the unlabeled instances in the target domain. In contrast, our instance weighting framework allows unlabeled target instances to contribute to the model estimation.

Blitzer et al. (2006) propose a domain adaptation method that uses the unlabeled target instances to infer a good feature representation, which can be regarded as weighting the features. In contrast, we weight the instances. The idea of using $\frac{p_t(x)}{p_s(x)}$ to weight instances has been studied in statistics (Shimodaira, 2000), but has not been applied to NLP tasks.

6 Conclusions and Future Work

Domain adaptation is a very important problem with applications to many NLP tasks. In this paper, we formally analyze the domain adaptation problem and propose a general instance weighting framework for domain adaptation. The framework is flexible to support many different strategies for adaptation. In particular, it can support adaptation with some target domain labeled instances as well as that without any labeled target instances. Experiment results on three NLP tasks show that while regular semi-supervised learning methods and supervised learning methods can be applied to domain adaptation without considering domain difference, they do not perform as well as our new method, which explicitly captures domain difference. Our results also show that incorporating and exploiting more information from the target domain is much more useful than excluding misleading training examples from the source domain. The framework opens up many interesting future research directions, especially those related to how to more accurately set/estimate those weighting parameters.

Acknowledgments

This work was in part supported by the National Science Foundation under award numbers 0425852 and 0428472. We thank the anonymous reviewers for their valuable comments.

References

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proc. of EMNLP*, pages 120–128.

Ciprian Chelba and Alex Acero. 2004. Adaptation of maximum entropy capitalizer: Little data can help a lot. In *Proc. of EMNLP*, pages 285–292.

Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *J. Artificial Intelligence Res.*, 26:101–126.

R. Florian, H. Hassan, A. Ittycheriah, H. Jing, N. Kambhatla, X. Luo, N. Nicolov, and S. Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *Proc. of HLT-NAACL*, pages 1–8.

Y. Grandvalet and Y. Bengio. 2005. Semi-supervised learning by entropy minimization. In *NIPS*.

Brian Roark and Michiel Bacchiani. 2003. Supervised and unsupervised PCFG adaptatin to novel domains. In *Proc. of HLT-NAACL*, pages 126–133.

Hidetoshi Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90:227–244.

The Infinite Tree

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning
Computer Science Department, Stanford University
Stanford, CA 94305
{jrfinkel, grenager, manning}@cs.stanford.edu

Abstract

Historically, unsupervised learning techniques have lacked a principled technique for selecting the number of unseen components. Research into non-parametric priors, such as the Dirichlet process, has enabled instead the use of *infinite models*, in which the number of hidden categories is not fixed, but can grow with the amount of training data. Here we develop the *infinite tree*, a new infinite model capable of representing recursive branching structure over an arbitrarily large set of hidden categories. Specifically, we develop three infinite tree models, each of which enforces different independence assumptions, and for each model we define a simple *direct assignment* sampling inference procedure. We demonstrate the utility of our models by doing unsupervised learning of part-of-speech tags from treebank dependency skeleton structure, achieving an accuracy of 75.34%, and by doing unsupervised splitting of part-of-speech tags, which increases the accuracy of a generative dependency parser from 85.11% to 87.35%.

1 Introduction

Model-based unsupervised learning techniques have historically lacked good methods for choosing the number of unseen components. For example, k-means or EM clustering require advance specification of the number of mixture components. But the introduction of nonparametric priors such as the *Dirichlet process* (Ferguson, 1973) enabled development of *infinite mixture models*, in which the number of hidden components is not fixed, but emerges naturally from the training data (Antoniak, 1974).

Teh et al. (2006) proposed the hierarchical Dirichlet process (HDP) as a way of applying the Dirichlet process (DP) to more complex model forms, so as to allow multiple, group-specific, infinite mixture models to *share* their mixture components. The closely related *infinite hidden Markov model* is an HMM in which the transitions are modeled using an HDP, enabling unsupervised learning of sequence models when the number of hidden states is unknown (Beal et al., 2002; Teh et al., 2006).

We extend this work by introducing the *infinite tree model*, which represents recursive branching structure over a potentially infinite set of hidden states. Such models are appropriate for the syntactic dependency structure of natural language. The hidden states represent word categories ("tags"), the observations they generate represent the words themselves, and the tree structure represents syntactic dependencies between pairs of tags.

To validate the model, we test unsupervised learning of tags conditioned on a given dependency tree structure. This is useful, because coarse-grained syntactic categories, such as those used in the Penn Treebank (PTB), make insufficient distinctions to be the basis of accurate syntactic parsing (Charniak, 1996). Hence, state-of-the-art parsers either supplement the part-of-speech (POS) tags with the lexical forms themselves (Collins, 2003; Charniak, 2000), manually split the tagset into a finer-grained one (Klein and Manning, 2003a), or learn finer grained tag distinctions using a heuristic learning procedure (Petrov et al., 2006). We demonstrate that the tags learned with our model are correlated with the PTB POS tags, and furthermore that they improve the accuracy of an automatic parser when used in training.

2 Finite Trees

We begin by presenting three *finite* tree models, each with different independence assumptions.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 272–279,
Prague, Czech Republic, June 2007.

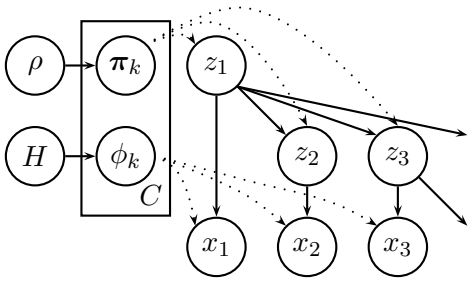


Figure 1: A graphical representation of the *finite* Bayesian tree model with independent children. The *plate* (rectangle) indicates that there is one copy of the model parameter variables for each state $k \leq C$.

2.1 Independent Children

In the first model, children are generated independently of each other, conditioned on the parent. Let t denote both the tree and its root node, $c(t)$ the list of children of t, $c_i(t)$ the i^{th} child of t, and $p(t)$ the parent of t. Each tree t has a hidden state z_t (in a syntax tree, the tag) and an observation x_t (the word).[1] The probability of a tree is given by the recursive definition:[2]

$$\mathrm{P}_{tr}(t) = \mathrm{P}(x_t|z_t) \prod_{t' \in c(t)} \mathrm{P}(z_{t'}|z_t) \mathrm{P}_{tr}(t')$$

To make the model Bayesian, we must define random variables to represent each of the model's parameters, and specify prior distributions for them. Let each of the hidden state variables have C possible values which we will index with k. Each state k has a distinct distribution over observations, parameterized by ϕ_k, which is distributed according to a prior distribution over the parameters H:

$$\phi_k|H \sim H$$

We generate each observation x_t from some distribution $F(\phi_{z_t})$ parameterized by ϕ_{z_t} specific to its corresponding hidden state z_t. If $F(\phi_k)$s are multinomials, then a natural choice for H would be a Dirichlet distribution.[3]

The hidden state $z_{t'}$ of each child is distributed according to a multinomial distribution π_{z_t} specific to the hidden state z_t of the parent:

$$x_t|z_t \sim F(\phi_{z_t})$$
$$z_{t'}|z_t \sim \text{Multinomial}(\pi_{z_t})$$

Each multinomial over children $\boldsymbol{\pi}_k$ is distributed according to a Dirichlet distribution with parameter ρ:

$$\boldsymbol{\pi}_k|\rho \sim \text{Dirichlet}(\rho, \ldots, \rho)$$

This model is presented graphically in Figure 1.

2.2 Simultaneous Children

The independent child model adopts strong independence assumptions, and we may instead want models in which the children are conditioned on more than just the parent's state. Our second model thus generates the states of all of the children $c(t)$ simultaneously:

$$\mathrm{P}_{tr}(t) = \mathrm{P}(x_t|z_t)\mathrm{P}((z_{t'})_{t' \in c(t)}|z_t) \prod_{t' \in c(t)} \mathrm{P}_{tr}(t')$$

where $(z_{t'})_{t' \in c(t)}$ indicates the list of tags of the children of t. To parameterize this model, we replace the multinomial distribution $\boldsymbol{\pi_k}$ over states with a multinomial distribution $\boldsymbol{\lambda_k}$ over lists of states.[4]

2.3 Markov Children

The very large domain size of the child lists in the simultaneous child model may cause problems of sparse estimation. Another alternative is to use a first-order Markov process to generate children, in which each child's state is conditioned on the previous child's state:

$$\mathrm{P}_{tr}(t) = \mathrm{P}(x_t|z_t) \prod_{i=1}^{|c(t)|} \mathrm{P}(z_{c_i(t)}|z_{c_{i-1}(t)}, z_t) \mathrm{P}_{tr}(t')$$

For this model, we augment all child lists with a distinguished *start node*, $c_0(t)$, which has as its state a distinguished *start state*, allowing us to capture the unique behavior of the first (observed) child. To parameterize this model, note that we will need to define $C(C+1)$ multinomials, one for each parent state and preceding child state (or a distinguished start state).

3 To Infinity, and Beyond ...

This section reviews needed background material for our approach to making our tree models infinite.

3.1 The Dirichlet Process

Suppose we model a document as a *bag of words* produced by a mixture model, where the mixture components might be *topics* such as business, politics, sports, etc. Using this model we can generate a

[1] To model length, every child list ends with a distinguished *stop node*, which has as its state a distinguished *stop state*.

[2] We also define a distinguished node t_0, which generates the root of the entire tree, and $P(x_{t_0}|z_{t_0}) = 1$.

[3] A Dirichlet *distribution* is a distribution over the possible parameters of a multinomial distributions, and is distinct from the Dirichlet *process*.

[4] This requires stipulating a maximum list length.

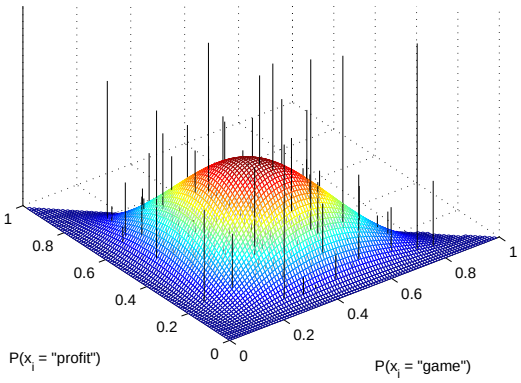


Figure 2: Plot of the density function of a Dirichlet distribution H (the surface) as well as a draw G (the vertical lines, or *sticks*) from a Dirichlet process $\text{DP}(\alpha_0, H)$ which has H as a base measure. Both distributions are defined over a simplex in which each point corresponds to a particular multinomial distribution over three possible words: "profit", "game", and "election". The placement of the sticks is drawn from the distribution H, and is independent of their lengths, which is drawn from a *stick-breaking* process with parameter α_0.

document by first generating a distribution over topics $\boldsymbol{\pi}$, and then for each position i in the document, generating a topic z_i from $\boldsymbol{\pi}$, and then a word x_i from the topic specific distribution ϕ_{z_i}. The word distributions ϕ_k for each topic k are drawn from a base distribution H. In Section 2, we sample C multinomials ϕ_k from H. In the infinite mixture model we sample an infinite number of multinomials from H, using the Dirichlet process.

Formally, given a base distribution H and a concentration parameter α_0 (loosely speaking, this controls the relative sizes of the topics), a Dirichlet process $\text{DP}(\alpha_0, H)$ is the distribution of a discrete random probability measure G over the same (possibly continuous) space that H is defined over; thus *it is a measure over measures*. In Figure 2, the sticks (vertical lines) show a draw G from a Dirichlet process where the base measure H is a Dirichlet distribution over 3 words. A draw comprises of an infinite number of sticks, and each corresponding topic.

We factor G into two coindexed distributions: $\boldsymbol{\pi}$, a distribution over the integers, where the integer represents the index of a particular topic (i.e., the height of the sticks in the figure represent the probability of the topic indexed by that stick) and $\boldsymbol{\phi}$, representing the word distribution of each of the topics (i.e., the location of the sticks in the figure). To generate $\boldsymbol{\pi}$ we first generate an infinite sequence of variables $\boldsymbol{\pi'} = (\pi'_k)_{k=1}^{\infty}$, each of which is distributed according to the Beta distribution:

$$\pi'_k | \alpha_0 \sim \text{Beta}(1, \alpha_0)$$

Then $\boldsymbol{\pi} = (\pi_k)_{k=1}^{\infty}$ is defined as:

$$\pi_k = \pi'_k \prod_{i=1}^{k-1} (1 - \pi'_i)$$

Following Pitman (2002) we refer to this process as $\pi \sim \text{GEM}(\alpha_0)$. It should be noted that $\sum_{k=1}^{\infty} \pi_k = 1$,[5] and $P(i) = \pi_i$. Then, according to the DP, $P(\phi_i) = \pi_i$. The complete model, is shown graphically in Figure 3(a).

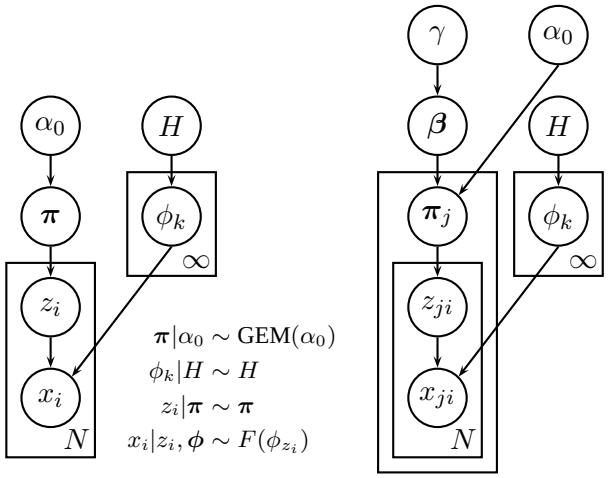


Figure 3: A graphical representation of a simple Dirichlet process mixture model (left) and a hierarchical Dirichlet process model (right). Note that we show the *stick-breaking* representations of the models, in which we have factored $G \sim \text{DP}(\alpha_0, H)$ into two sets of variables: π and ϕ.

To build intuition, we walk through the process of generating from the infinite mixture model for the document example, where x_i is the word at position i, and z_i is its topic. F is a multinomial distribution parameterized by $\boldsymbol{\phi}$, and H is a Dirichlet distribution. Instead of generating all of the infinite mixture components $(\pi_k)_{k=1}^{\infty}$ at once, we can build them up incrementally. If there are K known topics, we represent only the known elements $(\pi_k)_{k=1}^{K}$ and represent the remaining probability mass $\pi_u =$

[5] This is called the *stick-breaking* construction: we start with a stick of unit length, representing the entire probability mass, and successively break bits off the end of the stick, where the proportional amount broken off is represented by π'_k and the absolute amount is represented by π_k.

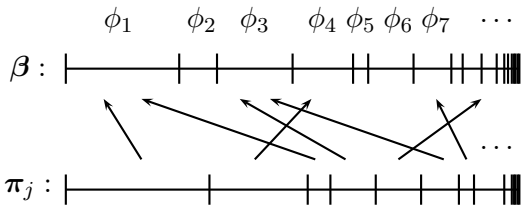


Figure 4: A graphical representation of $\boldsymbol{\pi}_j$, a broken stick, which is distributed according to a DP with a broken stick $\boldsymbol{\beta}$ as a base measure. Each β_k corresponds to a ϕ_k.

$1 - (\sum_{k=1}^{K} \pi_k)$. Initially we have $\pi_u = 1$ and $\boldsymbol{\phi} = ()$.

For the ith position in the document, we first draw a topic $z_i \sim \boldsymbol{\pi}$. If $z_i \neq u$, then we find the coindexed topic ϕ_{z_i}. If $z_i = u$, the unseen topic, we make a draw $b \sim \text{Beta}(1, \alpha_0)$ and set $\pi_{K+1} = b\pi_u$ and $\pi_u^{new} = (1-b)\pi_u$. Then we draw a parameter $\phi_{K+1} \sim H$ for the new topic, resulting in $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_{K+1}, \pi_u^{new})$ and $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_{K+1})$. A word is then drawn from this topic and emitted by the document.

3.2 The Hierarchical Dirichlet Process

Let's generalize our previous example to a corpus of documents. As before, we have a set of shared topics, but now each document has its *own characteristic distribution* over these topics. We represent topic distributions both locally (for each document) and globally (across all documents) by use of a hierarchical Dirichlet process (HDP), which has a local DP for each document, in which *the base measure is itself a draw from another, global, DP.*

The complete HDP model is represented graphically in Figure 3(b). Like the DP, it has global broken stick $\boldsymbol{\beta} = (\beta_k)_{k=1}^{\infty}$ and topic specific word distribution parameters $\boldsymbol{\phi} = (\phi_k)_{k=1}^{\infty}$, which are coindexed. It differs from the DP in that it also has local broken sticks $\boldsymbol{\pi}_j$ for each group j (in our case documents). While the global stick $\boldsymbol{\beta} \sim \text{GEM}(\gamma)$ is generated as before, the local sticks $\boldsymbol{\pi}_j$ are distributed according to a DP with base measure $\boldsymbol{\beta}$: $\boldsymbol{\pi}_j \sim \text{DP}(\alpha_0, \boldsymbol{\beta})$.

We illustrate this generation process in Figure 4. The upper unit line represents $\boldsymbol{\beta}$, where the size of segment k represents the value of element β_k, and the lower unit line represents $\boldsymbol{\pi}_j \sim \text{DP}(\alpha_0, \boldsymbol{\beta})$ for a particular group j. Each element of the lower stick was sampled from a particular element of the upper stick, and elements of the upper stick may be sampled multiple times or not at all; on average, larger elements will be sampled more often. Each element $\boldsymbol{\beta}_k$, as well as all elements of $\boldsymbol{\pi}_j$ that were sampled from it, corresponds to a particular ϕ_k. Critically, several distinct $\boldsymbol{\pi}_j$ can be sampled from the same β_k and hence share ϕ_k; *this is how components are shared among groups.*

For concreteness, we show how to generate a corpus of documents from the HDP, generating one document at a time, and incrementally constructing our infinite objects. Initially we have $\beta_u = 1$, $\boldsymbol{\phi} = ()$, and $\pi_{ju} = 1$ for all j. We start with the first position of the first document and draw a local topic $y_{11} \sim \boldsymbol{\pi}_1$, which will return u with probability 1. Because $y_{11} = u$ we must make a draw from the base measure, $\boldsymbol{\beta}$, which, because this is the first document, will also return u with probability 1. We must now break β_u into β_1 and β_u^{new}, and break π_{1u} into π_{11} and π_{1u}^{new} in the same manner presented for the DP. Since π_{11} now corresponds to global topic 1, we sample the word $x_{11} \sim \text{Multinomial}(\phi_1)$. To sample each subsequent word i, we first sample the local topic $y_{1i} \sim \boldsymbol{\pi}_1$. If $y_{1i} \neq u$, and $\pi_{1y_{1i}}$ corresponds to β_k in the global stick, then we sample the word $x_{1i} \sim \text{Multinomial}(\phi_k)$. Once the first document has been sampled, subsequent documents are sampled in a similar manner; initially $\pi_{ju} = 1$ for document j, while $\boldsymbol{\beta}$ continues to grow as more documents are sampled.

4 Infinite Trees

We now use the techniques from Section 3 to create infinite versions of each tree model from Section 2.

4.1 Independent Children

The changes required to make the Bayesian independent children model infinite don't affect its basic structure, as can be witnessed by comparing the graphical depiction of the infinite model in Figure 5 with that of the finite model in Figure 1. The instance variables z_t and x_t are parameterized as before. The primary change is that the number of copies of the state plate is infinite, as are the number of variables $\boldsymbol{\pi}_k$ and ϕ_k.

Note also that each distribution over possible child states $\boldsymbol{\pi}_k$ must also be infinite, since the number of possible child states is potentially infinite. We achieve this by representing each of the $\boldsymbol{\pi}_k$ variables as a broken stick, and adopt the same approach of

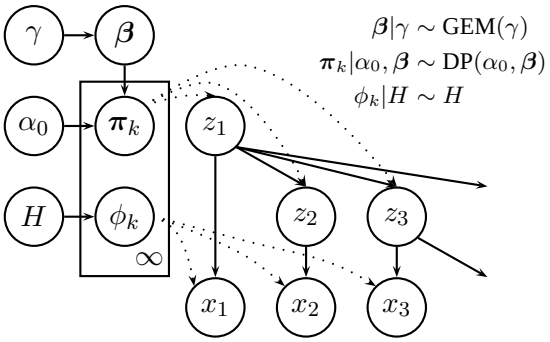


Figure 5: A graphical representation of the *infinite* independent child model.

sampling each $\boldsymbol{\pi}_k$ from a DP with base measure $\boldsymbol{\beta}$. For the dependency tree application, ϕ_k is a vector representing the parameters of a multinomial over words, and H is a Dirichlet distribution.

The infinite hidden Markov model (iHMM) or HDP-HMM (Beal et al., 2002; Teh et al., 2006) is a model of sequence data with transitions modeled by an HDP.[6] The iHMM can be viewed as a special case of this model, where each state (except the stop state) produces exactly one child.

4.2 Simultaneous Children

The key problem in the definition of the simultaneous children model is that of defining a distribution over the lists of children produced by each state, since each child in the list has as its domain the positive integers, representing the infinite set of possible states. Our solution is to construct a distribution L_k over lists of states from the distribution over individual states $\boldsymbol{\pi}_k$. The obvious approach is to sample the states at each position i.i.d.:

$$\text{P}((z_{t'})_{t' \in c(t)}|\boldsymbol{\pi}) = \prod_{t' \in c(t)} \text{P}(z_{t'}|\boldsymbol{\pi}) = \prod_{t' \in c(t)} \pi_{z_{t'}}$$

However, we want our model to be able to represent the fact that some child lists, $\boldsymbol{c}_t$, are more or less probable than the product of the individual child probabilities would indicate. To address this, we can sample a state-conditional distribution over child lists λ_k from a DP with L_k as a base measure. Thus, we augment the basic model given in the previous section with the variables ζ, L_k, and λ_k:

$$\begin{aligned} L_k|\boldsymbol{\pi}_k &\sim \text{Deterministic, as described above} \\ \lambda_k|\zeta, L_k &\sim \text{DP}(\zeta, L_k) \\ \boldsymbol{c}_t|\lambda_k &\sim \lambda_k \end{aligned}$$

An important consequence of defining L_k locally (instead of globally, using $\boldsymbol{\beta}$ instead of the $\boldsymbol{\pi}_k$s) is that the model captures not only what sequences of children a state prefers, but also the individual children that state prefers; if a state gives high probability to some particular sequence of children, then it is likely to also give high probability to other sequences containing those same states, or a subset thereof.

4.3 Markov Children

In the Markov children model, more copies of the variable $\boldsymbol{\pi}$ are needed, because each child state must be conditioned both on the parent state and on the state of the preceding child. We use a new set of variables $\boldsymbol{\pi}_{ki}$, where $\boldsymbol{\pi}$ is determined by the parent state k and the state of the preceding sibling i. Each of the $\boldsymbol{\pi}_{ki}$ is distributed as $\boldsymbol{\pi}_k$ was in the basic model: $\boldsymbol{\pi}_{ki} \sim \text{DP}(\alpha_0, \boldsymbol{\beta})$.

5 Inference

Our goal in inference is to draw a sample from the posterior over assignments of states to observations. We present an inference procedure for the infinite tree that is based on Gibbs sampling in the *direct assignment* representation, so named because we directly assign global state indices to observations.[7]

Before we present the procedure, we define a few count variables. Recall from Figure 4 that each state k has a local stick $\boldsymbol{\pi}_k$, each element of which corresponds to an element of $\boldsymbol{\beta}$. In our sampling procedure, we only keep elements of $\boldsymbol{\pi}_k$ and $\boldsymbol{\beta}$ which correspond to states observed in the data. We define the variable m_{jk} to be the number of elements of the finite observed portion of $\boldsymbol{\pi}_k$ which correspond to β_j and n_{jk} to be the number of observations with state k whose parent's state is j.

We also need a few model-specific counts. For the simultaneous children model we need $\overline{n}_{j\boldsymbol{z}}$, which is

[6] The original iHMM paper (Beal et al., 2002) predates, and was the motivation for, the work presented in Teh et al. (2006), and is the origin of the term *hierarchical Dirichlet process*. However, they used the term to mean something slightly different than the HDP presented in Teh et al. (2006), and presented a sampling scheme for inference that was a heuristic approximation of a Gibbs sampler.

[7] We adapt one of the sampling schemes mentioned by Teh et al. (2006) for use in the iHMM. This paper suggests two sampling schemes for inference, but does not explicitly present them. Upon discussion with one of the authors (Y. W. Teh, 2006, p.c.), it became clear that inference using the augmented representation is much more complicated than initially thought.

the number of times the state sequence $\boldsymbol{z}$ occurred as the children of state j. For the Markov children model we need the count variable $\hat{n}_{jik}$ which is the number of observations for a node with state k whose parent's state is j and whose previous sibling's state is i. In all cases we represent marginal counts using dot-notation, e.g., $n_{\cdot k}$ is the total number of nodes with state k, regardless of parent.

Our procedure alternates between three distinct sampling stages: (1) sampling the state assignments z, (2) sampling the counts m_{jk}, and (3) sampling the global stick $\boldsymbol{\beta}$. The only modification of the procedure that is required for the different tree models is the method for computing the probability of the child state sequence given the parent state $\mathrm{P}((z_{t'})_{t' \in c(t)}|z_t)$, defined separately for each model.

Sampling z. In this stage we sample a state for each tree node. The probability of node t being assigned state k is given by:

$$\mathrm{P}(z_t = k|\boldsymbol{z}^{-t}, \boldsymbol{\beta}) \propto \mathrm{P}(z_t = k, (z_{t'})_{t' \in s(t)}|z_{p(t)}) \cdot \mathrm{P}((z_{t'})_{t' \in c(t)}|z_t = k) \cdot f_k^{-x_t}(x_t)$$

where $s(t)$ denotes the set of siblings of t, $f_k^{-x_t}(x_t)$ denotes the posterior probability of observation x_t given all other observations assigned to state k, and $\boldsymbol{z}^{-t}$ denotes all state assignments except z_t. In other words, the probability is proportional to the product of three terms: the probability of the states of t and its siblings given its parent $z_{p(t)}$, the probability of the states of the children $c(t)$ given z_t, and the posterior probability of observation x_t given z_t. Note that if we sample z_t to be a previously unseen state, we will need to extend $\boldsymbol{\beta}$ as discussed in Section 3.2.

Now we give the equations for $\mathrm{P}((z_{t'})_{t' \in c(t)}|z_t)$ for each of the models. In the independent child model the probability of generating each child is:

$$\mathrm{P}_{\text{ind}}(z_{c_i(t)} = k|z_t = j) = \frac{n_{jk} + \alpha_0 \beta_k}{n_{j\cdot} + \alpha_0}$$

$$\mathrm{P}_{\text{ind}}((z_{t'})_{t' \in c(t)}|z_t = j) = \prod\nolimits_{t' \in c(t)} \mathrm{P}_{\text{ind}}(z_{t'}|z_t = j)$$

For the simultaneous child model, the probability of generating a sequence of children, $\mathbf{z}$, takes into account how many times that sequence has been generated, along with the likelihood of regenerating it:

$$\mathrm{P}_{\text{sim}}((z_{t'})_{t' \in c(t)} = \mathbf{z}|z_t = j) = \frac{\overline{n}_{j\mathbf{z}} + \zeta \mathrm{P}_{\text{ind}}(\mathbf{z}|z_t = j)}{\overline{n}_{j\cdot} + \zeta}$$

Recall that ζ denotes the concentration parameter for the sequence generating DP. Lastly, we have the Markov child model:

$$\mathrm{P}_{\text{m}}(z_{c_i(t)} = k|z_{c_{i-1}(t)} = i, z_t = j) = \frac{\hat{n}_{jik} + \alpha_0 \beta_k}{\hat{n}_{ji\cdot} + \alpha_0}$$

$$\mathrm{P}_{\text{m}}((z_{t'})_{t' \in c(t)}|z_t) = \prod\nolimits_{i=1}^{|c(t)|} \mathrm{P}_{\text{m}}(z_{c_i(t)}|z_{c_{i-1}(t)}, z_t)$$

Finally, we give the posterior probability of an observation, given that $F(\phi_k)$ is Multinomial(ϕ_k), and that H is Dirichlet$(\rho, \ldots, \rho)$. Let N be the vocabulary size and $\dot{n}_k$ be the number of observations x with state k. Then:

$$f_k^{-x_t}(x_t) = \frac{\dot{n}_{x_t k} + \rho}{\dot{n}_{\cdot k} + N\rho}$$

Sampling m. We use the following procedure, which slightly modifies one from (Y. W. Teh, 2006, p.c.), to sample each m_{jk}:

SAMPLEM(j, k)
1 **if** $n_{jk} = 0$
2 **then** $m_{jk} = 0$
3 **else** $m_{jk} = 1$
4 **for** $i \leftarrow 2$ **to** n_{jk}
5 **do if** rand() $< \frac{\alpha_0}{\alpha_0 + i - 1}$
6 **then** $m_{jk} = m_{jk} + 1$
7 **return** m_{jk}

Sampling β. Lastly, we sample $\boldsymbol{\beta}$ using the Dirichlet distribution:

$$(\beta_1, \ldots, \beta_K, \beta_u) \sim \text{Dirichlet}(m_{\cdot 1}, \ldots, m_{\cdot K}, \alpha_0)$$

Figure 6: An example of a syntactic dependency tree where the dependencies are between tags (hidden states), and each tag generates a word (observation).

6 Experiments

We demonstrate infinite tree models on two distinct syntax learning tasks: unsupervised POS learning conditioned on untagged dependency trees and learning a split of an existing tagset, which improves the accuracy of an automatic syntactic parser.

For both tasks, we use a simple modification of the basic model structure, to allow the trees to generate dependents on the left and the right with different distributions – as is useful in modeling natural language. The modification of the independent child tree is trivial: we have two copies of each of

the variables $\boldsymbol{\pi}_k$, one each for the left and the right. Generation of dependents on the right is completely independent of that for the left. The modifications of the other models are similar, but now there are separate *sets* of π_k variables for the Markov child model, and separate L_k and λ_k variables for the simultaneous child model, for each of the left and right.

For both experiments, we used dependency trees extracted from the Penn Treebank (Marcus et al., 1993) using the head rules and dependency extractor from Yamada and Matsumoto (2003). As is standard, we used WSJ sections 2–21 for training, section 22 for development, and section 23 for testing.

6.1 Unsupervised POS Learning

In the first experiment, we do unsupervised part-of-speech learning conditioned on dependency trees. To be clear, the input to our algorithm is the dependency structure skeleton of the corpus, but not the POS tags, and the output is a labeling of each of the words in the tree for word class. Since the model knows nothing about the POS annotation, the new classes have arbitrary integer names, and are not guaranteed to correlate with the POS tag definitions. We found that the choice of α_0 and β (the concentration parameters) did not affect the output much, while the value of ρ (the parameter for the base Dirichlet distribution) made a much larger difference. For all reported experiments, we set $\alpha_0 = \beta = 10$ and varied ρ.

We use several metrics to evaluate the word classes. First, we use the standard approach of greedily assigning each of the learned classes to the POS tag with which it has the greatest overlap, and then computing tagging accuracy (Smith and Eisner, 2005; Haghighi and Klein, 2006).[8] Additionally, we compute the mutual information of the learned clusters with the gold tags, and we compute the cluster F-score (Ghosh, 2003). See Table 1 for results of the different models, parameter settings, and metrics. Given the variance in the number of classes learned it is a little difficult to interpret these results, but it is clear that the Markov child model is the best; it achieves superior performance to the independent child model on all metrics, while learning fewer word classes. The poor performance of the simultaneous model warrants further investigation, but we observed that the distributions learned by that model are far more spiked, potentially due to double counting of tags, since the sequence probabilities are already based on the local probabilities.

[8]The advantage of this metric is that it's comprehensible. The disadvantage is that it's easy to inflate by adding classes.

Model	ρ	# Classes	Acc.	MI	F1
Indep.	0.01	943	67.89	2.00	48.29
	0.001	1744	73.61	2.23	40.80
	0.0001	2437	74.64	2.27	39.47
Simul.	0.01	183	21.36	0.31	21.57
	0.001	430	15.77	0.09	13.80
	0.0001	549	16.68	0.12	14.29
Markov	0.01	613	68.53	2.12	49.82
	0.001	894	75.34	2.31	48.73

Table 1: Results of part unsupervised POS tagging on the different models, using a greedy accuracy measure.

For comparison, Haghighi and Klein (2006) report an unsupervised baseline of 41.3%, and a best result of 80.5% from using hand-labeled prototypes and distributional similarity. However, they train on less data, and learn fewer word classes.

6.2 Unsupervised POS Splitting

In the second experiment we use the infinite tree models to learn a refinement of the PTB tags. We initialize the set of hidden states to the set of PTB tags, and then, during inference, constrain the sampling distribution over hidden state z_t at each node t to include only states that are a refinement of the annotated PTB tag at that position. The output of this training procedure is a new annotation of the words in the PTB with the learned tags. We then compare the performance of a generative dependency parser trained on the new refined tags with one trained on the base PTB tag set. We use the generative dependency parser distributed with the Stanford factored parser (Klein and Manning, 2003b) for the comparison, since it performs simultaneous tagging and parsing during testing. In this experiment, unlabeled, directed, dependency parsing accuracy for the best model increased from 85.11% to 87.35%, a 15% error reduction. See Table 2 for the full results over all models and parameter settings.

7 Related Work

The HDP-PCFG (Liang et al., 2007), developed at the same time as this work, aims to learn state splits for a binary-branching PCFG. It is similar to our simultaneous child model, but with several important distinctions. As discussed in Section 4.2, in our model each state has a DP over sequences, with a base distribution that is defined over the local child

Model	ρ	Accuracy
Baseline	–	85.11
Independent	0.01	86.18
	0.001	85.88
Markov	0.01	87.15
	0.001	87.35

Table 2: Results of untyped, directed dependency parsing, where the POS tags in the training data have been split according to the various models. At test time, the POS tagging and parsing are done simultaneously by the parser.

state probabilities. In contrast, Liang et al. (2007) define a global DP over sequences, with the base measure defined over the global state probabilities, β; locally, each state has an HDP, with this global DP as the base measure. We believe our choice to be more linguistically sensible: in our model, for a particular state, dependent sequences which are similar to one another increase one another's likelihood. Additionally, their modeling decision made it difficult to define a Gibbs sampler, and instead they use variational inference. Earlier, Johnson et al. (2007) presented *adaptor grammars*, which is a very similar model to the HDP-PCFG. However they did not confine themselves to a binary branching structure and presented a more general framework for defining the process for splitting the states.

8 Discussion and Future Work

We have presented a set of novel infinite tree models and associated inference algorithms, which are suitable for representing syntactic dependency structure. Because the models represent a potentially infinite number of hidden states, they permit unsupervised learning algorithms which naturally select a number of word classes, or tags, based on qualities of the data. Although they require substantial technical background to develop, the learning algorithms based on the models are actually simple in form, requiring only the maintenance of counts, and the construction of sampling distributions based on these counts. Our experimental results are preliminary but promising: they demonstrate that the model is capable of capturing important syntactic structure.

Much remains to be done in applying infinite models to language structure, and an interesting extension would be to develop inference algorithms that permit completely unsupervised learning of dependency structure.

Acknowledgments

Many thanks to Yeh Whye Teh for several enlightening conversations, and to the following members (and honorary member) of the Stanford NLP group for comments on an earlier draft: Thad Hughes, David Hall, Surabhi Gupta, Ani Nenkova, Sebastian Riedel. This work was supported by a Scottish Enterprise Edinburgh-Stanford Link grant (R37588), as part of the EASIE project, and by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Phase II Program.

References

C. E. Antoniak. 1974. Mixtures of Dirichlet processes with applications to Bayesian nonparametrics. *Annals of Statistics*, 2:1152–1174.

M.J. Beal, Z. Ghahramani, and C.E. Rasmussen. 2002. The infinite hidden Markov model. In *Advances in Neural Information Processing Systems*, pages 577–584.

E. Charniak. 1996. Tree-bank grammars. In *AAAI 1996*, pages 1031–1036.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *HLT-NAACL 2000*, pages 132–139.

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

T. S. Ferguson. 1973. A Bayesian analysis of some nonparametric problems. *Annals of Statistics*, 1:209–230.

J. Ghosh. 2003. Scalable clustering methods for data mining. In N. Ye, editor, *Handbook of Data Mining*, chapter 10, pages 247–277. Lawrence Erlbaum Assoc.

A. Haghighi and D. Klein. 2006. Prototype-driven learning for sequence models. In *HLT-NAACL 2006*.

M. Johnson, T. Griffiths, and S. Goldwater. 2007. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In *NIPS 2007*.

D. Klein and C. D. Manning. 2003a. Accurate unlexicalized parsing. In *ACL 2003*.

D. Klein and C. D. Manning. 2003b. Factored A* search for models over sequences and trees. In *IJCAI 2003*.

P. Liang, S. Petrov, D. Klein, and M. Jordan. 2007. Nonparametric PCFGs using Dirichlet processes. In *EMNLP 2007*.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL 44/COLING 21*, pages 433–440.

J. Pitman. 2002. Poisson-Dirichlet and GEM invariant distributions for split-and-merge transformations of an interval partition. *Combinatorics, Probability and Computing*, 11:501–514.

N. A. Smith and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *ACL 2005*.

Y. W. Teh, M.I. Jordan, M. J. Beal, and D.M. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101:1566–1581.

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

Guiding Semi-Supervision with Constraint-Driven Learning

Ming-Wei Chang **Lev Ratinov** **Dan Roth**
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{mchang21, ratinov2, danr}@uiuc.edu

Abstract

Over the last few years, two of the main research directions in machine learning of natural language processing have been the study of semi-supervised learning algorithms as a way to train classifiers when the labeled data is scarce, and the study of ways to exploit knowledge and global information in structured learning tasks. In this paper, we suggest a method for incorporating domain knowledge in semi-supervised learning algorithms. Our novel framework unifies and can exploit several kinds of *task specific constraints*. The experimental results presented in the information extraction domain demonstrate that applying constraints helps the model to generate better feedback during learning, and hence the framework allows for high performance learning with significantly less training data than was possible before on these tasks.

1 Introduction

Natural Language Processing (NLP) systems typically require large amounts of knowledge to achieve good performance. Acquiring labeled data is a difficult and expensive task. Therefore, an increasing attention has been recently given to semi-supervised learning, where large amounts of unlabeled data are used to improve the models learned from a small training set (Collins and Singer, 1999; Thelen and Riloff, 2002). The hope is that semi-supervised or even unsupervised approaches, when given enough knowledge about the *structure* of the problem, will be competitive with the supervised models trained on large training sets. However, in the general case, semi-supervised approaches give mixed results, and sometimes even degrade the model performance (Nigam et al., 2000). In many cases, improving semi-supervised models was done by *seeding* these models with domain information taken from dictionaries or ontology (Cohen and Sarawagi, 2004; Collins and Singer, 1999; Haghighi and Klein, 2006; Thelen and Riloff, 2002). On the other hand, in the supervised setting, it has been shown that incorporating domain and problem specific structured information can result in substantial improvements (Toutanova et al., 2005; Roth and Yih, 2005).

This paper proposes a novel constraints-based learning protocol for guiding semi-supervised learning. We develop a formalism for constraints-based learning that unifies several kinds of constraints: unary, dictionary based and n-ary constraints, which encode structural information and interdependencies among possible labels. One advantage of our formalism is that it allows capturing different levels of constraint violation. Our protocol can be used in the presence of any learning model, including those that acquire additional statistical constraints from observed data while learning (see Section 5. In the experimental part of this paper we use HMMs as the underlying model, and exhibit significant reduction in the number of training examples required in two information extraction problems.

As is often the case in semi-supervised learning, the algorithm can be viewed as a process that improves the model by generating feedback through

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 280–287,
Prague, Czech Republic, June 2007.

labeling unlabeled examples. Our algorithm pushes this intuition further, in that the use of constraints allows us to better exploit domain information as a way to label, along with the current learned model, unlabeled examples. Given a small amount of labeled data and a large unlabeled pool, our framework initializes the model with the labeled data and then repeatedly:
(1) Uses *constraints* and the learned model to label the instances in the pool.
(2) Updates the model by newly labeled data.

This way, we can generate *better* "training" examples during the semi-supervised learning process. The core of our approach, (1), is described in Section 5. The task is described in Section 3 and the Experimental study in Section 6. It is shown there that the improvement on the training examples via the constraints indeed boosts the learned model and the proposed method significantly outperforms the traditional semi-supervised framework.

2 Related Work

In the semi-supervised domain there are two main approaches for injecting domain specific knowledge. One is using the prior knowledge to accurately tailor the generative model so that it captures the domain structure. For example, (Grenager et al., 2005) proposes *Diagonal Transition Models* for sequential labeling tasks where neighboring words tend to have the same labels. This is done by constraining the HMM transition matrix, which can be done also for other models, such as CRF. However (Roth and Yih, 2005) showed that reasoning with more expressive, non-sequential constraints can improve the performance for the supervised protocol.

A second approach has been to use a small high-accuracy set of labeled tokens as a way to seed and bootstrap the semi-supervised learning. This was used, for example, by (Thelen and Riloff, 2002; Collins and Singer, 1999) in information extraction, and by (Smith and Eisner, 2005) in POS tagging. (Haghighi and Klein, 2006) extends the dictionary-based approach to sequential labeling tasks by propagating the information given in the seeds with contextual word similarity. This follows a conceptually similar approach by (Cohen and Sarawagi, 2004) that uses a large named-entity dictionary, where the similarity between the candidate named-entity and its matching prototype in the dictionary is encoded as a feature in a supervised classifier.

In our framework, dictionary lookup approaches are viewed as unary constraints on the output states. We extend these kinds of constraints and allow for more general, n-ary constraints.

In the supervised learning setting it has been established that incorporating global information can significantly improve performance on several NLP tasks, including information extraction and semantic role labeling. (Punyakanok et al., 2005; Toutanova et al., 2005; Roth and Yih, 2005). Our formalism is most related to this last work. But, we develop a semi-supervised learning protocol based on this formalism. We also make use of *soft* constraints and, furthermore, extend the notion of soft constraints to account for multiple levels of constraints' violation. Conceptually, although not technically, the most related work to ours is (Shen et al., 2005) that, in a somewhat ad-hoc manner uses soft constraints to guide an unsupervised model that was crafted for mention tracking. To the best of our knowledge, we are the first to suggest a general semi-supervised protocol that is driven by soft constraints.

We propose learning with constraints - a framework that combines the approaches described above in a unified and intuitive way.

3 Tasks, Examples and Datasets

In Section 4 we will develop a general framework for semi-supervised learning with constraints. However, it is useful to illustrate the ideas on concrete problems. Therefore, in this section, we give a brief introduction to the two domains on which we tested our algorithms. We study two information extraction problems in each of which, given text, a set of pre-defined fields is to be identified. Since the fields are typically related and interdependent, these kinds of applications provide a good test case for an approach like ours.[1]

The first task is to identify fields from citations (McCallum et al., 2000) . The data originally included 500 labeled references, and was later extended with 5,000 unannotated citations collected from papers found on the Internet (Grenager et al., 2005). Given a citation, the task is to extract the

[1]The data for both problems is available at: http://www.stanford.edu/ grenager/data/unsupie.tgz

(a) [*AUTHOR* Lars Ole Andersen .] [*TITLE* Program analysis and specialization for the C programming language .] [*TECH-REPORT* PhD thesis ,] [*INSTITUTION* DIKU , University of Copenhagen ,] [*DATE* May 1994 .]

(b) [*AUTHOR* Lars Ole Andersen . Program analysis and] [*TITLE* specialization for the] [*EDITOR* C] [*BOOKTITLE* Programming language] [*TECH-REPORT* . PhD thesis ,] [*INSTITUTION* DIKU , University of Copenhagen , May] [*DATE* 1994 .]

Figure 1: Error analysis of a HMM model. The labels are annotated by underline and are to the right of each open bracket. The correct assignment was shown in (a). While the predicted label assignment (b) is generally coherent, some constraints are violated. Most obviously, punctuation marks are ignored as cues for state transitions. The constraint "Fields cannot end with stop words (such as "the")" may be also good.

fields that appear in the given reference. See Fig. 1. There are 13 possible fields including author, title, location, etc.

To gain an insight to how the constraints can guide semi-supervised learning, assume that the sentence shown in Figure 1 appears in the unlabeled data pool. Part (a) of the figure shows the correct labeled assignment and part (b) shows the assignment labeled by a HMM trained on 30 labels. However, if we apply the constraint that state transition can occur only on punctuation marks, the same HMM model parameters will result in the correct labeling (a). Therefore, by adding the improved labeled assignment we can generate better training samples during semi-supervised learning. In fact, the punctuation marks are only some of the constraints that can be applied to this problem. The set of constraints we used in our experiments appears in Table 1. Note that some of the constraints are non-local and are very intuitive for people, yet it is very difficult to inject this knowledge into most models.

The second problem we consider is extracting fields from advertisements (Grenager et al., 2005). The dataset consists of 8,767 advertisements for apartment rentals in the San Francisco Bay Area downloaded in June 2004 from the Craigslist website. In the dataset, only 302 entries have been labeled with 12 fields, including *size, rent, neighborhood, features*, and so on. The data was preprocessed using regular expressions for phone numbers, email addresses and URLs. The list of the constraints for this domain is given in Table 1. We implement some global constraints and include unary constraints which were largely imported from the list of seed words used in (Haghighi and Klein, 2006). We slightly modified the seedwords due to difference in preprocessing.

4 Notation and Definitions

Consider a structured classification problem, where given an input sequence $x = (x_1, \ldots, x_N)$, the task is to find the best assignment to the output variables $y = (y_1, \ldots, y_M)$. We denote $\mathcal{X}$ to be the space of the possible input sequences and $\mathcal{Y}$ to be the set of possible output sequences.

We define a structured output classifier as a function $h : \mathcal{X} \rightarrow \mathcal{Y}$ that uses a global scoring function $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ to assign scores to each possible input/output pair. Given an input x, a desired function f will assign the correct output y the highest score among all the possible outputs. The global scoring function is often decomposed as a weighted sum of feature functions,

$$f(x, y) = \sum_{i=1}^{M} \lambda_i f_i(x, y) = \boldsymbol{\lambda} \cdot F(x, y).$$

This decomposition applies both to discriminative linear models and to generative models such as HMMs and CRFs, in which case the linear sum corresponds to log likelihood assigned to the input/output pair by the model (for details see (Roth, 1999) for the classification case and (Collins, 2002) for the structured case). Even when not dictated by the model, the feature functions $f_i(x, y)$ used are local to allow inference tractability. Local feature function can capture some context for each input or output variable, yet it is very limited to allow dynamic programming decoding during inference.

Now, consider a scenario where we have a set of constraints $C_1, \ldots, C_K$. We define a constraint $C : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}$ as a function that indicates whether the input/output sequence violates some desired properties. When the constraints are hard, the solution is given by

$$\operatorname*{argmax}_{y \in 1_{C(x)}} \boldsymbol{\lambda} \cdot F(x, y),$$

(a)-Citations

1) Each field must be a consecutive list of words, and can appear at most once in a citation.
2) State transitions must occur on punctuation marks.
3) The citation can only start with author or editor.
4) The words *pp., pages* correspond to *PAGE*.
5) Four digits starting with 20xx and 19xx are *DATE*.
6) Quotations can appear only in titles.
7) The words *note, submitted, appear* are *NOTE*.
8) The words *CA, Australia, NY* are *LOCATION*.
9) The words *tech, technical* are *TECH_REPORT*.
10) The words *proc, journal, proceedings, ACM* are *JOURNAL* or *BOOKTITLE*.
11) The words *ed, editors* correspond to *EDITOR*.

(b)-Advertisements

1) State transitions can occur only on punctuation marks or the newline symbol.
2) Each field must be at least 3 words long.
3) The words *laundry, kitchen, parking* are *FEATURES*.
4) The words *sq, ft, bdrm* are *SIZE*.
5) The word *$, *MONEY** are *RENT*.
6) The words *close, near, shopping* are *NEIGHBORHOOD*.
7) The words *laundry kitchen, parking* are *FEATURES*.
8) The (normalized) words *phone, email* are *CONTACT*.
9) The words *immediately, begin, cheaper* are *AVAILABLE*.
10) The words *roommates, respectful, drama* are *ROOMMATES*.
11) The words *smoking, dogs, cats* are *RESTRICTIONS*.
12) The word *http, image, link* are *PHOTOS*.
13) The words *address, carlmont, st, cross* are *ADDRESS*.
14) The words *utilities, pays, electricity* are *UTILITIES*.

Table 1: The list of constraints for extracting fields from citations and advertisements. Some constraints (represented in the first block of each domain) are global and are relatively difficult to inject into traditional models. While all the constraints hold for the vast majority of the data, some of them are violated by some correct labeled assignments.

where $1_{C(x)}$ is a subset of $\mathcal{Y}$ for which all C_i assign the value 1 for the given (x, y).

When the constraints are soft, we want to incur some penalty for their violation. Moreover, we want to incorporate into our cost function a measure for the *amount* of violation incurred by violating the constraint. A generic way to capture this intuition is to introduce a distance function $d(y, 1_{C_i(x)})$ between the space of outputs that respect the constraint,$1_{C_i(x)}$, and the given output sequence y. One possible way to implement this distance function is as the minimal Hamming distance to a sequence that respects the constraint C_i, that is: $d(y, 1_{C_i(x)}) = \min_{(y' \in 1_{C(x)})} H(y, y')$. If the penalty for violating the soft constraint C_i is ρ_i, we write the score function as:

$$\operatorname*{argmax}_y \boldsymbol{\lambda} \cdot F(x, y) - \sum_{i=1}^{K} \rho_i d(y, 1_{C_i(x)}) \qquad (1)$$

We refer to $d(y, 1_{C(x)})$ as the *valuation* of the constraint C on (x, y). The intuition behind (1) is as follows. Instead of merely maximizing the model's likelihood, we also want to bias the model using some knowledge. The first term of (1) is used to learn from data. The second term biases the mode by using the knowledge encoded in the constraints. Note that we do not normalize our objective function to be a true probability distribution.

5 Learning and Inference with Constraints

In this section we present a new constraint-driven learning algorithm (**CODL**) for using constraints to guide semi-supervised learning. The task is to learn the parameter vector $\boldsymbol{\lambda}$ by using the new objective function (1). While our formulation allows us to train also the coefficients of the constraints valuation, ρ_i, we choose not to do it, since we view this as a way to bias (or enforce) the prior knowledge into the learned model, rather than allowing the data to brush it away. Our experiments demonstrate that the proposed approach is robust to inaccurate approximation of the prior knowledge (assigning the same penalty to all the ρ_i).

We note that in the presence of constraints, the inference procedure (for finding the output y that maximizes the cost function) is usually done with search techniques (rather than Viterbi decoding, see (Toutanova et al., 2005; Roth and Yih, 2005) for a discussion), we chose beamsearch decoding.

The semi-supervised learning with constraints is done with an EM-like procedure. We initialize the model with traditional supervised learning (ignoring the constraints) on a small labeled set. Given an unlabeled set U, in the estimation step, the traditional EM algorithm assigns a distribution over labeled assignments $\mathcal{Y}$ of each $x \in U$, and in the maximization step, the set of model parameters is learned from the distributions assigned in the estimation step.

However, in the presence of constraints, assigning the complete distributions in the estimation step is infeasible since the constraints reshape the distribution in an arbitrary way. As in existing methods for training a model by maximizing a linear cost function (maximize likelihood or discriminative maxi-

mization), the distribution over $\mathcal{Y}$ is represented as the set of scores assigned to it; rather than considering the score assigned to *all* $y's$, we truncate the distribution to the top K assignments as returned by the search. Given a set of K top assignments $y^1, \ldots, y^K$, we approximate the estimation step by assigning uniform probability to the top K candidates, and zero to the other output sequences. We denote this algorithm *top-K hard EM*. In this paper, we use beamsearch to generate K candidates according to (1).

Our training algorithm is summarized in Figure 2. Several things about the algorithm should be clarified: the Top-K-Inference procedure in line 7, the learning procedure in line 9, and the new parameter estimation in line 9.

The Top-K-Inference is a procedure that returns the K labeled assignments that maximize the new objective function (1). In our case we used the top-K elements in the beam, but this could be applied to any other inference procedure. The fact that the constraints are used in the inference procedure (in particular, for generating new training examples) allows us to use a learning algorithm that ignores the constraints, which is a lot more efficient (although algorithms that do take the constraints into account can be used too). We used maximum likelihood estimation of $\boldsymbol{\lambda}$ but, in general, perceptron or quasi-Newton can also be used.

It is known that traditional semi-supervised training can degrade the learned model's performance. (Nigam et al., 2000) has suggested to balance the contribution of labeled and unlabeled data to the parameters. The intuition is that when iteratively estimating the parameters with EM, we disallow the parameters to drift too far from the supervised model. The parameter re-estimation in line 9, uses a similar intuition, but instead of weighting data instances, we introduced a smoothing parameter γ which controls the convex combination of models induced by the labeled and the unlabeled data. Unlike the technique mentioned above which focuses on naive Bayes, our method allows us to weight linear models generated by different learning algorithms.

Another way to look the algorithm is from the self-training perspective (McClosky et al., 2006). Similarly to self-training, we use the current model to generate new training examples from the unlabeled set. However, there are two important differences. One is that in self-training, once an unlabeled sample was labeled, it is never labeled again. In our case all the samples are relabeled in each iteration. In self-training it is often the case that only high-confidence samples are added to the labeled data pool. While we include all the samples in the training pool, we could also limit ourselves to the high-confidence samples. The second difference is that each unlabeled example generates K labeled instances. The case of one iteration of *top-1 hard EM* is equivalent to self training, where all the unlabeled samples are added to the labeled pool.

Input:
$Cycles$: learning cycles
$Tr = \{x, y\}$: labeled training set.
U: unlabeled dataset
F: set of feature functions.
$\{\rho_i\}$: set of penalties.
$\{C_i\}$: set of constraints.
γ: balancing parameter with the supervised model.
$learn(Tr, F)$: supervised learning algorithm
Top-K-Inference:
returns top-K labeled scored by the cost function (1)

CODL:
1. Initialize $\boldsymbol{\lambda}_0 = learn(Tr, F)$.
2. $\boldsymbol{\lambda} = \boldsymbol{\lambda}_0$.
3. For $Cycles$ iterations do:
4. $T = \phi$
5. For each $x \in U$
6. $\{(x, y^1), \ldots, (x, y^K)\} =$
7. Top-K-Inference$(x, \boldsymbol{\lambda}, F, \{C_i\}, \{\rho_i\})$
8. $T = T \cup \{(x, y^1), \ldots, (x, y^K)\}$
9. $\boldsymbol{\lambda} = \gamma\boldsymbol{\lambda}_0 + (1 - \gamma)learn(T, F)$

Figure 2: **CO**nstraint **D**riven **L**earning (CODL). In Top-K-Inference, we use beamsearch to find the K-best solution according to Eq. (1).

There are several possible benefits to using $K > 1$ samples. (1) It effectively increases the training set by a factor of K (albeit by somewhat noisy examples). In the structured scenario, each of the top-K assignments is likely to have some good components so generating top-K assignments helps leveraging the noise. (2) Given an assignment that does not satisfy some constraints, using top-K allows for multiple ways to correct it. For example, consider the output 11101000 with the constraint that it should belong to the language 1^*0^*. If the two top scoring corrections are 11111000 and 11100000, considering only one of those can negatively bias the model.

6 Experiments and Results

In this section, we present empirical results of our algorithms on two domains: *citations* and *advertisements*. Both problems are modeled with a simple token-based HMM. We stress that token-based HMM cannot represent many of our constraints. The function $d(y, 1_{C(x)})$ used is an approximation of a Hamming distance function, discussed in Section 7. For both domains, and all the experiments, γ was set to 0.1. The constraints violation penalty ρ is set to $-\log 10^{-4}$ and $-\log 10^{-1}$ for citations and advertisements, resp.[2] Note that all constraints share the same penalty. The number of semi-supervised training cycles (line 3 of Figure 2) was set to 5. The constraints for the two domains are listed in Table 1.

We trained models on training sets of size varying from 5 to 300 for the citations and from 5 to 100 for the advertisements. Additionally, in all the semi-supervised experiments, 1000 unlabeled examples are used. We report token-based[3] accuracy on 100 held-out examples (which do not overlap neither with the training nor with the unlabeled data). We ran 5 experiments in each setting, randomly choosing the training set. The results reported below are the averages over these 5 runs.

To verify our claims we implemented several baselines. The first baseline is the supervised learning protocol denoted by **sup**. The second baseline was a traditional **top-1 Hard EM** also known as truncated EM[4] (denoted by **H** for Hard). In the third baseline, denoted **H&W**, we balanced the weight of the supervised and unsupervised models as described in line 9 of Figure 2. We compare these baselines to our proposed protocol, **H&W&C**, where we added the constraints to guide the **H&W** protocol. We experimented with two flavors of the algorithm: the top-1 and the top-K version. In the top-K version, the algorithm uses K-best predictions (K=50) for each instance in order to update the model as described in Figure 2.

The experimental results for both domains are in given Table 2. As hypothesized, hard EM sometimes degrade the performance. Indeed, with 300 labeled examples in the citations domain, the performance decreases from 86.1 to 80.7. The usefulness of injecting constraints in semi-supervised learning is exhibited in the two right most columns: using constraints **H&W&C** improves the performance over **H&W** quite significantly.

(a)- Citations

N	Inf.	**sup.**	**H**	**H&W**	**H&W&C** (Top-1)	**H&W&C** (Top-K)
5	no I	55.1	60.9	63.6	70.6	71.0
	I	66.6	69.0	72.5	76.0	77.8
10	no I	64.6	66.8	69.8	76.5	76.7
	I	78.1	78.1	81.0	83.4	83.8
15	no I	68.7	70.6	73.7	78.6	79.4
	I	81.3	81.9	84.1	85.5	86.2
20	no I	70.1	72.4	75.0	79.6	79.4
	I	81.1	82.4	84.0	86.1	86.1
25	no I	72.7	73.2	77.0	81.6	82.0
	I	84.3	84.2	86.2	87.4	87.6
300	no I	86.1	80.7	87.1	88.2	88.2
	I	92.5	89.6	93.4	93.6	93.5

(b)-Advertisements

N	Inf.	**sup.**	**H**	**H&W**	**H&W&C** (Top-1)	**H&W&C** (Top-K)
5	no I	55.2	61.8	60.5	66.0	66.0
	I	59.4	65.2	63.6	69.3	69.6
10	no I	61.6	69.2	67.0	70.8	70.9
	I	66.6	73.2	71.6	74.7	74.7
15	no I	66.3	71.7	70.1	73.0	73.0
	I	70.4	75.6	74.5	76.6	76.9
20	no I	68.1	72.8	72.0	74.5	74.6
	I	71.9	76.7	75.7	77.9	78.1
25	no I	70.0	73.8	73.0	74.9	74.8
	I	73.7	77.7	76.6	78.4	78.5
100	no I	76.3	76.2	77.6	78.5	78.6
	I	80.4	80.5	81.2	81.8	81.7

Table 2: Experimental results for extracting fields from citations and advertisements. N is the number of labeled samples. **H** is the traditional hard-EM and **H&W** weighs labeled and unlabeled data as mentioned in Sec. 5. Our proposed model is **H&W&C**, which uses constraints in the learning procedure. **I** refers to using constraints during inference at evaluation time. Note that adding constraints improves the accuracy during both learning and inference.

We carefully examined the contribution of using constraints to the learning stage and the testing stage, and two separate results are presented: testing with constraints (denoted *I* for inference) and without constraints (*no I*). The *I* results are consistently better. And, it is also clear from Table 2, that using constraints in training always improves

[2] The guiding intuition is that $\boldsymbol{\lambda} F(x, y)$ corresponds to a log-likelihood of a HMM model and ρ to a crude estimation of the log probability that a constraint does not hold. ρ was tuned on a development set and kept fixed in all experiments.

[3] Each token (word or punctuation mark) is assigned a state.

[4] We also experimented with (soft) EM without constraints, but the results were generally worse.

the model and the amount of improvement depends on the amount of labeled data.

Figure 3 compares two protocols on the advertisements domain: **H&W+I**, where we first run the **H&W** protocol and then apply the constraints during testing stage, and **H&W&C+I**, which uses constraints to guide the model during learning and uses it also in testing. Although injecting constraints in the learning process helps, testing with constraints is more important than using constraints during learning, especially when the labeled data size is large. This confirms results reported for the supervised learning case in (Punyakanok et al., 2005; Roth and Yih, 2005). However, as shown, our proposed algorithm **H&W&C** for *training with constraints* is critical when the amount labeled data is small.

Figure 4 further strengthens this point. In the citations domain, **H&W&C+I** achieves with 20 labeled samples similar performance to the supervised version without constraints with 300 labeled samples.

(Grenager et al., 2005) and (Haghighi and Klein, 2006) also report results for semi-supervised learning for these domains. However, due to different preprocessing, the comparison is not straightforward. For the citation domain, when 20 labeled and 300 unlabeled samples are available, (Grenager et al., 2005) observed an increase from 65.2% to 71.3%. Our improvement is from 70.1% to 79.4%. For the advertisement domain, they observed no improvement, while our model improves from 68.1% to 74.6% with 20 labeled samples. Moreover, we successfully use out-of-domain data (web data) to improve our model, while they report that this data did not improve their unsupervised model.

(Haghighi and Klein, 2006) also worked on one of our data sets. Their underlying model, Markov Random Fields, allows more expressive features. Nevertheless, when they use only unary constraints they get 53.75%. When they use their final model, along with a mechanism for extending the prototypes to other tokens, they get results that are comparable to our model with 10 labeled examples. Additionally, in their framework, it is not clear how to use small amounts of labeled data when available. Our model outperforms theirs once we add 10 more examples.

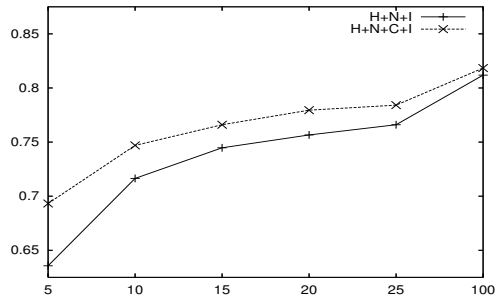


Figure 3: Comparison between **H&W+I** and **H&W&C+I** on the *advertisements* domain. When there is a lot of labeled data, inference with constraints is more important than using constraints during learning. However, it is important to train with constraints when the amount of labeled data is small.

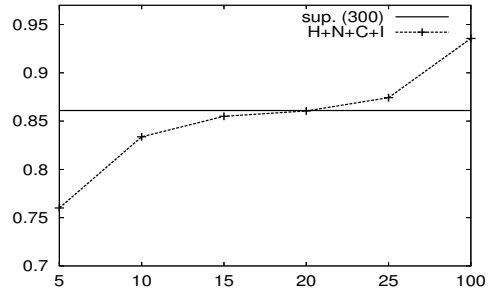


Figure 4: With 20 labeled citations, our algorithm performs competitively to the supervised version trained on 300 samples.

7 Soft Constraints

This section discusses the importance of using soft constraints rather than hard constraints, the choice of Hamming distance for $d(y, 1_{C(x)})$ and how we approximate it. We use two constraints to illustrate the ideas. *(C_1):* "state transitions can only occur on punctuation marks or newlines", and *(C_2):* "the field *TITLE* must appear".

First, we claim that defining $d(y, 1_{C(x)})$ to be the Hamming distance is superior to using a binary value, $d(y, 1_{C(x)}) = 0$ if $y \in 1_{C(x)}$ and 1 otherwise. Consider, for example, the constraint C_1 in the advertisements domain. While the vast majority of the instances satisfy the constraint, some violate it in more than one place. Therefore, once the binary distance is set to 1, the algorithm looses the ability to discriminate constraint violations in other locations

of the same instance. This may hurt the performance in both the inference and the learning stage.

Computing the Hamming distance exactly can be a computationally hard problem. Furthermore, it is unreasonable to implement the exact computation for each constraint. Therefore, we implemented a generic approximation for the hamming distance assuming only that we are given a boolean function $\phi_C(y_N)$ that returns whether labeling the token x_N with state y_N violates constraint with respect to an already labeled sequence $(x_1, \ldots, x_{N-1}, y_1, \ldots, y_{N-1})$. Then $d(y, 1_{C(x)}) = \sum_{i=1}^{N} \phi_C(y_i)$. For example, consider the prefix x_1, x_2, x_3, x_4, which contains no punctuation or newlines and was labeled $AUTH, AUTH, DATE, DATE$. This labeling violates C_1, the minimal hamming distance is 2, and our approximation gives 1, (since there is only one transition that violates the constraint.)

For constraints which cannot be validated based on prefix information, our approximation resorts to binary violation count. For instance, the constraint C_2 cannot be implemented with prefix information when the assignment is not complete. Otherwise, it would mean that the field TITLE should appear as early as possible in the assignment.

While (Roth and Yih, 2005) showed the significance of using hard constraints, our experiments show that using soft constraints is a superior option. For example, in the advertisements domain, C_1 holds for the large majority of the gold-labeled instances, but is sometimes violated. In supervised training with 100 labeled examples on this domain, **sup** gave 76.3% accuracy. When the constraint violation penalty ρ was *infinity* (equivalent to hard constraint), the accuracy improved to 78.7%, but when the penalty was set to $-log(0.1)$, the accuracy of the model jumped to 80.6%.

8 Conclusions and Future Work

We proposed to use constraints as a way to guide semi-supervised learning. The framework developed is general both in terms of the representation and expressiveness of the constraints, and in terms of the underlying model being learned – HMM in the current implementation. Moreover, our framework is a useful tool when the domain knowledge cannot be expressed by the model.

The results show that constraints improve not only the performance of the final inference stage but also propagate useful information during the semi-supervised learning process and that training with the constraints is especially significant when the number of labeled training data is small.

Acknowledgments: This work is supported by NSF SoD-HCER-0613885 and by a grant from Boeing. Part of this work was done while Dan Roth visited the Technion, Israel, supported by a Lady Davis Fellowship.

References

W. Cohen and S. Sarawagi. 2004. Exploiting dictionaries in named entity extraction: Combining semi-markov extraction processes and data integration methods. In *Proc. of the ACM SIGKDD*.

M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *Proc. of EMNLP*.

M. Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP*.

T. Grenager, D. Klein, and C. Manning. 2005. Unsupervised learning of field segmentation models for information extraction. In *Proc. of the Annual Meeting of the ACL*.

A. Haghighi and D. Klein. 2006. Prototype-driven learning for sequence models. In *Proc. of HTL-NAACL*.

A. McCallum, D. Freitag, and F. Pereira. 2000. Maximum entropy markov models for information extraction and segmentation. In *Proc. of ICML*.

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

K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. 2005. Learning and inference over constrained output. In *Proc. of IJCAI*.

D. Roth and W. Yih. 2005. Integer linear programming inference for conditional random fields. In *Proc. of ICML*.

D. Roth. 1999. Learning in natural language. In *Proc. of IJCAI*, pages 898–904.

W. Shen, X. Li, and A. Doan. 2005. Constraint-based entity matching. In *Proc. of AAAI)*.

N. Smith and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. of the Annual Meeting of the ACL*.

M. Thelen and E. Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proc. of EMNLP*.

K. Toutanova, A. Haghighi, and C. D. Manning. 2005. Joint learning improves semantic role labeling. In *Proc. of the Annual Meeting of the ACL*.

Supertagged Phrase-Based Statistical Machine Translation

Hany Hassan
School of Computing,
Dublin City University,
Dublin 9, Ireland
hhasan@computing.dcu.ie

Khalil Sima'an
Language and Computation,
University of Amsterdam,
Amsterdam, The Netherlands
simaan@science.uva.nl

Andy Way
School of Computing,
Dublin City University,
Dublin 9, Ireland
away@computing.dcu.ie

Abstract

Until quite recently, extending Phrase-based Statistical Machine Translation (PBSMT) with syntactic structure caused system performance to deteriorate. In this work we show that incorporating lexical syntactic descriptions in the form of supertags can yield significantly better PBSMT systems. We describe a novel PBSMT model that integrates supertags into the target language model and the target side of the translation model. Two kinds of supertags are employed: those from Lexicalized Tree-Adjoining Grammar and Combinatory Categorial Grammar. Despite the differences between these two approaches, the supertaggers give similar improvements. In addition to supertagging, we also explore the utility of a surface global grammaticality measure based on combinatory operators. We perform various experiments on the Arabic to English NIST 2005 test set addressing issues such as sparseness, scalability and the utility of system subcomponents. Our best result (0.4688 BLEU) improves by 6.1% relative to a state-of-the-art PBSMT model, which compares very favourably with the leading systems on the NIST 2005 task.

1 Introduction

Within the field of Machine Translation, by far the most dominant paradigm is Phrase-based Statistical Machine Translation (PBSMT) (Koehn et al., 2003; Tillmann & Xia, 2003). However, unlike in rule- and example-based MT, it has proven difficult to date to incorporate linguistic, syntactic knowledge in order to improve translation quality. Only quite recently have (Chiang, 2005) and (Marcu et al., 2006) shown that incorporating some form of syntactic structure could show improvements over a baseline PBSMT system. While (Chiang, 2005) avails of structure which is not linguistically motivated, (Marcu et al., 2006) employ syntactic structure to enrich the entries in the phrase table.

In this paper we explore a novel approach towards extending a standard PBSMT system with syntactic descriptions: we inject *lexical* descriptions into both the target side of the phrase translation table and the target language model. Crucially, the kind of lexical descriptions that we employ are those that are commonly devised within lexicon-driven approaches to linguistic syntax, e.g. Lexicalized Tree-Adjoining Grammar (Joshi & Schabes, 1992; Bangalore & Joshi, 1999) and Combinary Categorial Grammar (Steedman, 2000). In these linguistic approaches, it is assumed that the grammar consists of a very rich lexicon and a tiny, *impoverished*[1] set of combinatory operators that assemble lexical entries together into parse-trees. The lexical entries consist of syntactic constructs ('supertags') that describe information such as the POS tag of the word, its subcategorization information and the hierarchy of phrase categories that the word projects upwards. In this work we employ the lexical entries but exchange the algebraic combinatory operators with the more robust

[1] These operators neither carry nor presuppose further linguistic knowledge beyond what the lexicon contains.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 288–295,
Prague, Czech Republic, June 2007.

and efficient *supertagging* approach: like standard taggers, supertaggers employ probabilities based on local context and can be implemented using finite state technology, e.g. Hidden Markov Models (Bangalore & Joshi, 1999).

There are currently two supertagging approaches available: LTAG-based (Bangalore & Joshi, 1999) and CCG-based (Clark & Curran, 2004). Both the LTAG (Chen et al., 2006) and the CCG supertag sets (Hockenmaier, 2003) were acquired from the WSJ section of the Penn-II Treebank using hand-built extraction rules. Here we test both the LTAG and CCG supertaggers. We interpolate (log-linearly) the supertagged components (language model and phrase table) with the components of a standard PBSMT system. Our experiments on the Arabic–English NIST 2005 test suite show that each of the supertagged systems significantly improves over the baseline PBSMT system. Interestingly, combining the two taggers together diminishes the benefits of supertagging seen with the individual LTAG and CCG systems. In this paper we discuss these and other empirical issues.

The remainder of the paper is organised as follows: in section 2 we discuss the related work on enriching PBSMT with syntactic structure. In section 3, we describe the baseline PBSMT system which our work extends. In section 4, we detail our approach. Section 5 describes the experiments carried out, together with the results obtained. Section 6 concludes, and provides avenues for further work.

2 Related Work

Until very recently, the experience with adding syntax to PBSMT systems was negative. For example, (Koehn et al., 2003) demonstrated that adding syntax actually harmed the quality of their SMT system. Among the first to demonstrate improvement when adding recursive structure was (Chiang, 2005), who allows for hierarchical phrase probabilities that handle a range of reordering phenomena in the correct fashion. Chiang's derived grammar does not rely on any linguistic annotations or assumptions, so that the 'syntax' induced is not linguistically motivated.

Coming right up to date, (Marcu et al., 2006) demonstrate that 'syntactified' target language phrases can improve translation quality for Chinese–English. They employ a stochastic, top-down transduction process that assigns a joint probability to a source sentence and each of its alternative translations when rewriting the target parse-tree into a source sentence. The rewriting/transduction process is driven by "xRS rules", each consisting of a pair of a source phrase and a (possibly only partially) lexicalized syntactified target phrase. In order to extract xRS rules, the word-to-word alignment induced from the parallel training corpus is used to guide heuristic tree 'cutting' criteria.

While the research of (Marcu et al., 2006) has much in common with the approach proposed here (such as the syntactified target phrases), there remain a number of significant differences. Firstly, rather than induce millions of xRS rules from parallel data, we extract phrase pairs in the standard way (Och & Ney, 2003) and associate with each phrase-pair a set of target language syntactic structures based on supertag sequences. Relative to using arbitrary parse-chunks, the power of supertags lies in the fact that they are, syntactically speaking, rich lexical descriptions. A supertag can be assigned to every word in a phrase. On the one hand, the correct sequence of supertags could be assembled together, using only impoverished combinatory operators, into a small set of constituents/parses ('almost' a parse). On the other hand, because supertags are lexical entries, they facilitate robust syntactic processing (using Markov models, for instance) which does not necessarily aim at building a fully connected graph.

A second major difference with xRS rules is that our supertag-enriched target phrases *need not* be generalized into (xRS or any other) rules that work with abstract categories. Finally, like POS tagging, supertagging is more efficient than actual parsing or tree transduction.

3 Baseline Phrase-Based SMT System

We present the baseline PBSMT model which we extend with supertags in the next section. Our baseline PBSMT model uses GIZA++[2] to obtain word-level alignments in both language directions. The bidirectional word alignment is used to obtain phrase translation pairs using heuristics presented in

[2] http://www.fjoch.com/GIZA++.html

(Och & Ney, 2003) and (Koehn et al., 2003), and the Moses decoder was used for phrase extraction and decoding.[3]

Let t and s be the target and source language sentences respectively. Any (target or source) sentence x will consist of two parts: a bag of elements (words/phrases etc.) and an order over that bag. In other words, $x = \langle \phi_x, O_x \rangle$, where ϕ_x stands for the bag of phrases that constitute x, and O_x for the order of the phrases as given in x (O_x can be implemented as a function from a bag of tokens ϕ_x to a set with a finite number of positions). Hence, we may separate order from content:

$$\arg\max_t P(t|s) = \arg\max_t P(s \mid t)P(t) \quad (1)$$

$$= \arg\max_{\langle \phi_t, O_t \rangle} \overbrace{P(\phi_s \mid \phi_t)}^{TM} \overbrace{P(O_s \mid O_t)}^{distortion} \overbrace{P_w(t)}^{LM} \quad (2)$$

Here, $P_w(t)$ is the target language model, $P(O_s|O_t)$ represents the conditional (order) linear distortion probability, and $P(\phi_s|\phi_t)$ stands for a probabilistic translation model from target language bags of phrases to source language bags of phrases using a phrase translation table. As commonly done in PB-SMT, we interpolate these models log-linearly (using different λ weights) together with a word penalty weight which allows for control over the length of the target sentence t:

$$\arg\max_{\langle \phi_t, O_t \rangle} P(\phi_s \mid \phi_t)\ P(O_s \mid O_t)^{\lambda_o}$$
$$P_w(t)^{\lambda_{lm}}\ exp^{|t|\lambda_w}$$

For convenience of notation, the interpolation factor for the bag of phrases translation model is shown in formula (3) at the phrase level (but that does not entail any difference). For a bag of phrases ϕ_t consisting of phrases t_i, and bag ϕ_s consisting of phrases s_i, the phrase translation model is given by:

$$P(\phi_s \mid \phi_t) = \prod_{\substack{s_i \\ t_i}} P(s_i|t_i)$$

$$P(s_i|\ t_i) = P_{ph}(s_i|t_i)^{\lambda_{t1}} P_w(s_i|t_i)^{\lambda_{t2}} P_r(t_i|s_i)^{\lambda_{t3}} \quad (3)$$

where P_{ph} and P_r are the phrase-translation probability and its reverse probability, and P_w is the lexical translation probability.

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

4 Our Approach: Supertagged PBSMT

We extend the baseline model with lexical linguistic representations (*supertags*) both in the language model as well as in the phrase translation model. Before we describe how our model extends the baseline, we shortly review the supertagging approaches in Lexicalized Tree-Adjoining Grammar and Combinatory Categorial Grammar.

4.1 Supertags: Lexical Syntax

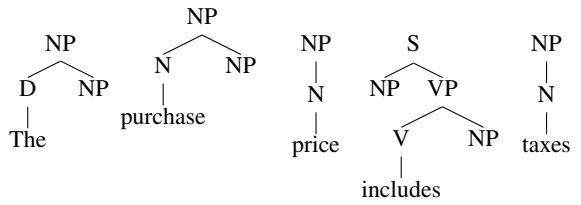


Figure 1: An LTAG supertag sequence for the sentence *The purchase price includes taxes*. The subcategorization information is most clearly available in the verb *includes* which takes a subject NP to its left and an object NP to its right.

Modern linguistic theory proposes that a syntactic parser has access to an extensive lexicon of word-structure pairs and a small, impoverished set of operations to manipulate and combine the lexical entries into parses. Examples of formal instantiations of this idea include CCG and LTAG. The lexical entries are syntactic constructs (graphs) that specify information such as POS tag, subcategorization/dependency information and other syntactic constraints at the level of agreement features. One important way of portraying such lexical descriptions is via the supertags devised in the LTAG and CCG frameworks (Bangalore & Joshi, 1999; Clark & Curran, 2004).

A supertag (see Figure 1) represents a complex, linguistic word category that encodes a syntactic structure expressing a specific local behaviour of a word, in terms of the arguments it takes (e.g. subject, object) and the syntactic environment in which it appears. In fact, in LTAG a supertag is an elementary tree and in CCG it is a CCG lexical category. Both descriptions can be viewed as closely related functional descriptions.

The term "supertagging" (Bangalore & Joshi, 1999) refers to tagging the words of a sentence, each

with a supertag. When well-formed, an ordered sequence of supertags can be viewed as a compact representation of a small set of constituents/parses that can be obtained by assembling the supertags together using the appropriate combinatory operators (such as substitution and adjunction in LTAG or function application and combination in CCG). Akin to POS tagging, the process of *supertagging* an input utterance proceeds with statistics that are based on the probability of a word-supertag pair given their Markovian or local context (Bangalore & Joshi, 1999; Clark & Curran, 2004). This is the main difference with full parsing: supertagging the input utterance need not result in a fully connected graph.

The LTAG-based supertagger of (Bangalore & Joshi, 1999) is a standard HMM tagger and consists of a (second-order) Markov language model over supertags and a lexical model conditioning the probability of every word on its own supertag (just like standard HMM-based POS taggers).

The CCG supertagger (Clark & Curran, 2004) is based on log-linear probabilities that condition a supertag on features representing its context. The CCG supertagger does not constitute a language model nor are the Maximum Entropy estimates directly interpretable as such. In our model we employ the CCG supertagger to obtain the best sequences of supertags for a corpus of sentences from which we obtain language model statistics. Besides the difference in probabilities and statistical estimates, these two supertaggers differ in the way the supertags are extracted from the Penn Treebank, cf. (Hockenmaier, 2003; Chen et al., 2006). Both supertaggers achieve a supertagging accuracy of 90–92%.

Three aspects make supertags attractive in the context of SMT. Firstly, supertags are rich syntactic constructs that exist for individual words and so they are easy to integrate into SMT models that can be based on any level of granularity, be it word- or phrase-based. Secondly, supertags specify the local syntactic constraints for a word, which resonates well with sequential (finite state) statistical (e.g. Markov) models. Finally, because supertags are rich lexical descriptions that represent underspecification in parsing, it is possible to have some of the benefits of full parsing without imposing the strict connectedness requirements that it demands.

4.2 A Supertag-Based SMT model

We employ the aforementioned supertaggers to enrich the English side of the parallel training corpus with a single supertag sequence per sentence. Then we extract phrase-pairs together with the cooccuring English supertag sequence from this corpus via the same phrase extraction method used in the baseline model. This way we directly extend the baseline model described in section 3 with supertags both in the phrase translation table and in the language model. Next we define the probabilistic model that accompanies this syntactic enrichment of the baseline model.

Let ST represent a supertag sequence of the same length as a target sentence t. Equation (2) changes as follows:

$$\arg\max_{t} \sum_{ST} P(s \mid t, ST) P_{ST}(t, ST) \approx$$

$$\arg\max_{\langle t, ST \rangle} \overbrace{P(\phi_s \mid \phi_{t,ST})}^{TM\ w.sup.tags} \overbrace{P(O_s \mid O_t)^{\lambda_o}}^{distortion}$$

$$\overbrace{P_{ST}(t, ST)}^{LM\ w.sup.tags} \overbrace{exp^{|t|\lambda_w}}^{word-penalty}$$

The approximations made in this formula are of two kinds: the standard split into components and the search for the most likely joint probability of a target hypothesis and a supertag sequence cooccuring with the source sentence (a kind of Viterbi approach to avoid the complex optimization involving the sum over supertag sequences). The distortion and word penalty models are the same as those used in the baseline PBSMT model.

Supertagged Language Model The 'language model' $P_{ST}(t, ST)$ is a supertagger assigning probabilities to sequences of word–supertag pairs. The language model is further smoothed by log-linear interpolation with the baseline language model over word sequences.

Supertags in Phrase Tables The supertagged phrase translation probability consists of a combination of supertagged components analogous to their counterparts in the baseline model (equation (3)), i.e. it consists of $P(s \mid t, ST)$, its reverse and a word-level probability. We smooth this probability by log-linear interpolation with the factored

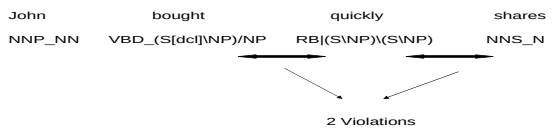


Figure 2: Example CCG operator violations: $V = 2$ and $L = 3$, and so the penalty factor is $1/3$.

backoff version $P(s \mid t)P(s \mid ST)$, where we import the baseline phrase table probability and exploit the probability of a source phrase given the target supertag sequence. A model in which we omit $P(s \mid ST)$ turns out to be slightly less optimal than this one.

As in most state-of-the-art PBSMT systems, we use GIZA++ to obtain word-level alignments in both language directions. The bidirectional word alignment is used to obtain lexical phrase translation pairs using heuristics presented in (Och & Ney, 2003) and (Koehn et al., 2003). Given the collected phrase pairs, we estimate the phrase translation probability distribution by relative frequency as follows:

$$\hat{P}_{ph}(s|t) = \frac{count(s,t)}{\sum_s count(s,t)}$$

For each extracted lexical phrase pair, we extract the corresponding supertagged phrase pairs from the supertagged target sequence in the training corpus (cf. section 5). For each lexical phrase pair, there is at least one corresponding supertagged phrase pair. The probability of the supertagged phrase pair is estimated by relative frequency as follows:

$$P_{st}(s|t, st) = \frac{count(s,t,st)}{\sum_s count(s,t,st)}$$

4.3 LMs with a Grammaticality Factor

The supertags usually encode dependency information that could be used to construct an 'almost parse' with the help of the CCG/LTAG composition operators. The *n*-gram language model over supertags applies a kind of statistical 'compositionality check' but due to smoothing effects this could mask crucial violations of the compositionality operators of the grammar formalism (CCG in this case). It is interesting to observe the effect of integrating into the language model a penalty imposed when formal compostion operators are violated. We combine the *n*-gram language model with a penalty factor that measures the number of encountered combinatory operator violations in a sequence of supertags (cf. Figure 2). For a supertag sequence of length (L) which has (V) operator violations (as measured by the CCG system), the language model P will be adjusted as $P* = P \times (1 - \frac{V}{L})$. This is of course no longer a simple smoothed maximum-likelihood estimate nor is it a true probability. Nevertheless, this mechanism provides a simple, efficient integration of a global compositionality (grammaticality) measure into the *n*-gram language model over supertags.

Decoder The decoder used in this work is Moses, a log-linear decoder similar to Pharaoh (Koehn, 2004), modified to accommodate supertag phrase probabilities and supertag language models.

5 Experiments

In this section we present a number of experiments that demonstrate the effect of lexical syntax on translation quality. We carried out experiments on the NIST open domain news translation task from Arabic into English. We performed a number of experiments to examine the effect of supertagging approaches (CCG or LTAG) with varying data sizes.

Data and Settings The experiments were conducted for Arabic to English translation and tested on the NIST 2005 evaluation set. The systems were trained on the LDC Arabic–English parallel corpus; we use the news part (130K sentences, about 5 million words) to train systems with what we call the *small* data set, and the news and a large part of the UN data (2 million sentences, about 50 million words) for experiments with *large* data sets.

The *n*-gram target language model was built using 250M words from the English GigaWord Corpus using the SRILM toolkit.[4] Taking 10% of the English GigaWord Corpus used for building our target language model, the supertag-based target language models were built from 25M words that were supertagged. For the LTAG supertags experiments, we used the LTAG English supertagger[5] (Bangalore

[4] http://www.speech.sri.com/projects/srilm/

[5] http://www.cis.upenn.edu/~xtag/gramrelease.html

& Joshi, 1999) to tag the English part of the parallel data and the supertag language model data. For the CCG supertag experiments, we used the CCG supertagger of (Clark & Curran, 2004) and the Edinburgh CCG tools[6] to tag the English part of the parallel corpus as well as the CCG supertag language model data.

The NIST MT03 test set is used for development, particularly for optimizing the interpolation weights using Minimum Error Rate training (Och, 2003).

Baseline System The baseline system is a state-of-the-art PBSMT system as described in section 3. We built two baseline systems with two different-sized training sets: 'Base-SMALL' (5 million words) and 'Base-LARGE' (50 million words) as described above. Both systems use a trigram language model built using 250 million words from the English GigaWord Corpus. Table 1 presents the BLEU scores (Papineni et al., 2002) of both systems on the NIST 2005 MT Evaluation test set.

System	BLEU Score
Base-SMALL	0.4008
Base-LARGE	0.4418

Table 1: Baseline systems' BLEU scores

5.1 Baseline vs. Supertags on Small Data Sets

We compared the translation quality of the baseline systems with the LTAG and CCG supertags systems (LTAG-SMALL and CCG-SMALL). The results are given in Table 2. All systems were trained on the same parallel data. The LTAG supertag-based system outperforms the baseline by 1.97 BLEU points absolute (or 4.9% relative), while the CCG supertag-based system scores 1.66 BLEU points over the baseline (4.1% relative). These significant improvements indicate that the rich information in supertags helps select better translation candidates.

System	BLEU Score
Base-SMALL	0.4008
LTAG-SMALL	0.4205
CCG-SMALL	0.4174

Table 2: LTAG and CCG systems on small data

POS Tags vs. Supertags A supertag is a complex tag that localizes the dependency and the syntax information from the context, whereas a normal POS tag just describes the general syntactic category of the word without further constraints. In this experiment we compared the effect of using supertags and POS tags on translation quality. As can be seen in Table 3, while the POS tags help (0.65 BLEU points, or 1.7% relative increase over the baseline), they clearly underperform compared to the supertag model (by 3.2%).

System	BLEU Score
Base-SMALL	0.4008
POS-SMALL	0.4073
LTAG-SMALL	.0.4205

Table 3: Comparing the effect of supertags and POS tags

The Usefulness of a Supertagged LM In these experiments we study the effect of the two added feature (cost) functions: supertagged translation and language models. We compare the baseline system to the supertags system with the supertag phrase-table probability but without the supertag LM. Table 4 lists the baseline system (Base-SMALL), the LTAG system without supertagged language model (LTAG-TM-ONLY) and the LTAG-SMALL system with both supertagged translation and language models. The results presented in Table 4 indicate that the improvement is a shared contribution between the supertagged translation and language models: adding the LTAG TM improves BLEU score by 1.38 points (3.4% relative) over the baseline, with the LTAG LM improving BLEU score by

System	BLEU Score
Base-SMALL	0.4008
LTAG-TM-ONLY	0.4146
LTAG-SMALL	.0.4205

Table 4: The effect of supertagged components

[6] http://groups.inf.ed.ac.uk/ccg/software.html

a further 0.59 points (a further 1.4% increase).

5.2 Scalability: Larger Training Corpora

Outperforming a PBSMT system on small amounts of training data is less impressive than doing so on really large sets. The issue here is scalability as well as whether the PBSMT system is able to bridge the performance gap with the supertagged system when reasonably large sizes of training data are used. To this end, we trained the systems on 2 million sentences of parallel data, deploying LTAG supertags and CCG supertags. Table 5 presents the comparison between these systems and the baseline trained on the same data. The LTAG system improves by 1.17 BLEU points (2.6% relative), but the CCG system gives an even larger increase: 1.91 BLEU points (4.3% relative). While this is slightly lower than the 4.9% relative improvement with the smaller data sets, the sustained increase is probably due to observing more data with different supertag contexts, which enables the model to select better target language phrases.

System	BLEU Score
Base-LARGE	0.4418
LTAG-LARGE	0.4535
CCG-LARGE	**0.4609**

Table 5: The effect of more training data

Adding a grammaticality factor As described in section 4.3, we integrate an impoverished grammaticality factor based on two standard CCG combination operations, namely Forward and Backward Application. Table 6 compares the results of the baseline, the CCG with an n-gram LM-only system (CCG-LARGE) and CCG-LARGE with this 'grammaticalized' LM system (CCG-LARGE-GRAM). We see that bringing the grammaticality tests to bear onto the supertagged system gives a further improvement of 0.79 BLEU points, a 1.7% relative increase, culminating in an overall increase of 2.7 BLEU points, or a 6.1% relative improvement over the baseline system.

System	BLEU Score
Base-LARGE	0.4418
CCG-LARGE	0.4609
CCG-LARGE-GRAM	**0.4688**

Table 6: Comparing the effect of CCG-GRAM

5.3 Discussion

A natural question to ask is whether LTAG and CCG supertags are playing similar (overlapping, or conflicting) roles in practice. Using an oracle to choose the best output of the two systems gives a BLEU score of 0.441, indicating that the combination provides significant room for improvement (cf. Table 2). However, our efforts to build a system that benefits from the combination using a simple log-linear combination of the two models did not give any significant performance change relative to the baseline CCG system. Obviously, more informed ways of combining the two could result in better performance than a simple log-linear interpolation of the components.

Figure 3 shows some example system output. While the baseline system omits the verb giving "the authorities that it had...", both the LTAG and CCG found a formulation "authorities reported that" with a closer meaning to the reference translation "The authorities said that". Omitting verbs turns out to be a problem for the baseline system when translating the notorious verbless Arabic sentences (see Figure 4). The supertagged systems have a more grammatically strict language model than a standard word-level Markov model, thereby exhibiting a preference (in the CCG system especially) for the insertion of a verb with a similar meaning to that contained in the reference sentence.

6 Conclusions

SMT practitioners have on the whole found it difficult to integrate syntax into their systems. In this work, we have presented a novel model of PBSMT which integrates supertags into the target language model and the target side of the translation model.

Using LTAG supertags gives the best improvement over a state-of-the-art PBSMT system for a smaller data set, while CCG supertags work best on a large 2 million-sentence pair training set. Adding grammaticality factors based on algebraic compositional operators gives the best result, namely 0.4688 BLEU, or a 6.1% relative increase over the baseline.

Reference: *The authorities said he was allowed to contact family members by phone from the armored vehicle he was in.*
Baseline: *the authorities that it had allowed him to communicate by phone with his family of the armored car where*
LTAG: *authorities reported that it had allowed him to contact by telephone with his family of armored car where*
CCG: *authorities reported that it had enabled him to communicate by phone his family members of the armored car where*

Figure 3: Sample output from different systems

Source*: wmn AlmErwf An Al$Eb AlSyny mHb llslAm .* **Ref***: It is well known that the Chinese people are peace loving .*
Baseline: *It is known that the Chinese people a peace-loving .*
LTAG*: It is known that the Chinese people a peace loving .* **CCG***: It is known that the Chinese people are peace loving .*

Figure 4: Verbless Arabic sentence and sample output from different systems

This result compares favourably with the best systems on the NIST 2005 Arabic–English task. We expect more work on system integration to improve results still further, and anticipate that similar increases are to be seen for other language pairs.

Acknowledgements

We would like to thank Srinivas Bangalore and the anonymous reviewers for useful comments on earlier versions of this paper. This work is partially funded by Science Foundation Ireland Principal Investigator Award 05/IN/1732, and Netherlands Organization for Scientific Research (NWO) VIDI Award.

References

S. Bangalore and A. Joshi, "Supertagging: An Approach to Almost Parsing", *Computational Linguistics* **25**(2):237–265, 1999.

J. Chen, S. Bangalore, and K. Vijay-Shanker, "Automated extraction of tree-adjoining grammars from treebanks". *Natural Language Engineering*, 12(3):251–299, 2006.

D. Chiang, "A Hierarchical Phrase-Based Model for Statistical Machine Translation", in *Proceedings of ACL 2005*, Ann Arbor, MI., pp.263–270, 2005.

S. Clark and J. Curran, "The Importance of Supertagging for Wide-Coverage CCG Parsing", in *Proceedings of COLING-04*, Geneva, Switzerland, pp.282–288, 2004.

J. Hockenmaier, *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*, PhD thesis, University of Edinburgh, UK, 2003.

A. Joshi and Y. Schabes, "Tree Adjoining Grammars and Lexicalized Grammars" in M. Nivat and A. Podelski (eds.) *Tree Automata and Languages*, Amsterdam, The Netherlands: North-Holland, pp.409–431, 1992.

P. Koehn, "Pharaoh: A Beam Search Decoder for phrase-based Statistical Machine Translation Models", in *Proceedings of AMTA-04*, Berlin/Heidelberg, Germany: Springer Verlag, pp.115–124, 2004.

P. Koehn, F. Och, and D. Marcu, "Statistical Phrase-Based Translation", in *Proceedings of HLT-NAACL 2003*, Edmonton, Canada, pp.127–133, 2003.

D. Marcu, W. Wang, A. Echihabi and K. Knight, "SPMT: Statistical Machine Translation with Syntactified Target Language Phrases", in *Proceedings of EMNLP*, Sydney, Australia, pp.44–52, 2006.

D. Marcu and W. Wong, "A Phrase-Based, Joint Probability Model for Statistical Machine Translation", in *Proceedings of EMNLP*, Philadelphia, PA., pp.133–139, 2002.

F. Och, "Minimum Error Rate Training in Statistical Machine Translation", in *Proceedings of ACL 2003*, Sapporo, Japan, pp.160–167, 2003.

F. Och and H. Ney, "A Systematic Comparison of Various Statistical Alignment Models", *Computational Linguistics* **29**:19–51, 2003.

K. Papineni, S. Roukos, T. Ward and W-J. Zhu, "BLEU: A Method for Automatic Evaluation of Machine Translation", in *Proceedings of ACL 2002*, Philadelphia, PA., pp.311–318, 2002.

L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", in A. Waibel & F-K. Lee (eds.) *Readings in Speech Recognition*, San Mateo, CA.: Morgan Kaufmann, pp.267–296, 1990.

M. Steedman, *The Syntactic Process*. Cambridge, MA: The MIT Press, 2000.

C. Tillmann and F. Xia, "A Phrase-based Unigram Model for Statistical Machine Translation", in *Proceedings of HLT-NAACL 2003*, Edmonton, Canada. pp.106–108, 2003.

Regression for Sentence-Level MT Evaluation with Pseudo References

Joshua S. Albrecht and Rebecca Hwa
Department of Computer Science
University of Pittsburgh
{jsa8,hwa}@cs.pitt.edu

Abstract

Many automatic evaluation metrics for machine translation (MT) rely on making comparisons to human translations, a resource that may not always be available. We present a method for developing sentence-level MT evaluation metrics that do not directly rely on human reference translations. Our metrics are developed using regression learning and are based on a set of weaker indicators of fluency and adequacy (*pseudo references*). Experimental results suggest that they rival standard reference-based metrics in terms of correlations with human judgments on new test instances.

1 Introduction

Automatic assessment of translation quality is a challenging problem because the evaluation task, at its core, is based on subjective human judgments. Reference-based metrics such as BLEU (Papineni et al., 2002) have rephrased this subjective task as a somewhat more objective question: how closely does the translation resemble sentences that are known to be good translations for the same source? This approach requires the participation of human translators, who provide the "gold standard" reference sentences. However, keeping humans in the evaluation loop represents a significant expenditure both in terms of time and resources; therefore it is worthwhile to explore ways of reducing the degree of human involvement.

To this end, Gamon et al. (2005) proposed a learning-based evaluation metric that does not compare against reference translations. Under a learning framework, the input (i.e., the sentence to be evaluated) is represented as a set of *features*. These are measurements that can be extracted from the input sentence (and may be individual metrics themselves). The learning algorithm combines the features to form a model (a composite evaluation metric) that produces the final score for the input. Without human references, the features in the model proposed by Gamon et al. were primarily language model features and linguistic indicators that could be directly derived from the input sentence alone. Although their initial results were not competitive with standard reference-based metrics, their studies suggested that a referenceless metric may still provide useful information about translation fluency. However, a potential pitfall is that systems might "game the metric" by producing fluent outputs that are not adequate translations of the source.

This paper proposes an alternative approach to evaluate MT outputs without comparing against human references. While our metrics are also trained, our model consists of different features and is trained under a different learning regime. Crucially, our model includes features that capture some notions of adequacy by comparing the input against *pseudo references*: sentences from other MT systems (such as commercial off-the-shelf systems or open sourced research systems). To improve fluency judgments, the model also includes features that compare the input against target-language "references" such as large text corpora and treebanks.

Unlike human translations used by standard reference-based metrics, pseudo references are not

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 296–303,
Prague, Czech Republic, June 2007.

"gold standards" and can be worse than the sentences being evaluated; therefore, these "references" in-and-of themselves are not necessarily informative enough for MT evaluation. The main insight of our approach is that through regression, the trained metrics can make more nuanced comparisons between the input and pseudo references. More specifically, our regression objective is to infer a function that maps a feature vector (which measures an input's similarity to the pseudo references) to a score that indicates the quality of the input. This is achieved by optimizing the model's output to correlate against a set of training examples, which are translation sentences labeled with quantitative assessments of their quality by human judges. Although this approach does incur some human effort, it is primarily for the development of training data, which, ideally, can be amortized over a long period of time.

To determine the feasibility of the proposed approach, we conducted empirical studies that compare our trained metrics against standard reference-based metrics. We report three main findings. First, pseudo references are informative comparison points. Experimental results suggest that a regression-trained metric that compares against pseudo references can have higher correlations with human judgments than applying standard metrics with multiple human references. Second, the learning model that uses both adequacy and fluency features performed the best, with adequacy being the more important factor. Third, when the pseudo references are multiple MT systems, the regression-trained metric is predictive even when the input is from a better MT system than those providing the references. We conjecture that comparing MT outputs against other imperfect translations allows for a more nuanced discrimination of quality.

2 Background and Related Work

For a formally organized event, such as the annual MT Evaluation sponsored by National Institute of Standard and Technology (NIST MT Eval), it may be worthwhile to recruit multiple human translators to translate a few hundred sentences for evaluation references. However, there are situations in which multiple human references are not practically available (e.g., the source may be of a large quantity, and no human translation exists). One such instance is translation quality assurance, in which one wishes to identify poor outputs in a large body of machine translated text automatically for human to post-edit. Another instance is in day-to-day MT research and development, where new test set with multiple references are also hard to come by. One could work with previous datasets from events such as the NIST MT Evals, but there is a danger of over-fitting. One also could extract a single reference from parallel corpora, although it is known that automatic metrics are more reliable when comparing against multiple references.

The aim of this work is to develop a trainable automatic metric for evaluation without human references. This can be seen as a form of confidence estimation on MT outputs (Blatz et al., 2003; Ueffing et al., 2003; Quirk, 2004). The main distinction is that confidence estimation is typically performed with a particular system in mind, and may rely on system-internal information in estimation. In this study, we draw on only system-independent indicators so that the resulting metric may be more generally applied. This allows us to have a clearer picture of the contributing factors as they interact with different types of MT systems.

Also relevant is previous work that applied machine learning approaches to MT evaluation, both with human references (Corston-Oliver et al., 2001; Kulesza and Shieber, 2004; Albrecht and Hwa, 2007; Liu and Gildea, 2007) and without (Gamon et al., 2005). One motivation for the learning approach is the ease of combining multiple criteria. Literature in translation evaluation reports a myriad of criteria that people use in their judgments, but it is not clear how these factors should be combined mathematically. Machine learning offers a principled and unified framework to induce a computational model of human's decision process. Disparate indicators can be encoded as one or more input features, and the learning algorithm tries to find a mapping from input features to a score that quantifies the input's quality by optimizing the model to match human judgments on training examples. The framework is attractive because its objective directly captures the goal of MT evaluation: how would a user rate the quality of these translations?

This work differs from previous approaches in

two aspects. One is the representation of the model; our model treats the metric as a distance measure even though there are no human references. Another is the training of the model. More so than when human references are available, regression is central to the success of the approach, as it determines how much we can trust the distance measures against each pseudo reference system.

While our model does not use human references directly, its features are adapted from the following distance-based metrics. The well-known BLEU (Papineni et al., 2002) is based on the number of common n-grams between the translation hypothesis and human reference translations of the same sentence. Metrics such as ROUGE, Head Word Chain (HWC), METEOR, and other recently proposed methods all offer different ways of comparing machine and human translations. ROUGE utilizes 'skip n-grams', which allow for matches of sequences of words that are not necessarily adjacent (Lin and Och, 2004a). METEOR uses the Porter stemmer and synonym-matching via WordNet to calculate recall and precision more accurately (Banerjee and Lavie, 2005). The HWC metrics compare dependency and constituency trees for both reference and machine translations (Liu and Gildea, 2005).

3 MT Evaluation with Pseudo References using Regression

Reference-based metrics are typically thought of as measurements of "similarity to good translations" because human translations are used as references, but in more general terms, they are distance measurements between two sentences. The distance between a translation hypothesis and an imperfect reference is still somewhat informative. As a toy example, consider a one-dimensional line segment. A distance from the end-point uniquely determines the position of a point. When the reference location is anywhere else on the line segment, a relative distance to the reference does not uniquely specify a location on the line segment. However, the position of a point can be uniquely determined if we are given its relative distances to two reference locations.

The problem space for MT evaluation, though more complex, is not dissimilar to the toy scenario. There are two main differences. First, we do not know the actual distance function – this is what we are trying to learn. The distance functions we have at our disposal are all heuristic approximations to the true translational distance function. Second, unlike human references, whose quality value is assumed to be maximum, the quality of a pseudo reference sentence is not known. In fact, prior to training, we do not even know the quality of the reference systems. Although the direct way to calibrate a reference system is to evaluate *its* outputs, this is not practically ideal, since human judgments would be needed each time we wish to incorporate a new reference system. Our proposed alternative is to calibrate the reference systems against an existing set of human judgments for a range of outputs from different MT systems. That is, if many of the reference system's outputs are similar to those MT outputs that received low assessments, we conclude this reference system may not be of high quality. Thus, if a new translation is found to be similar with this reference system's output, it is more likely for the new translation to also be bad.

Both issues of combining evidences from heuristic distances and calibrating the quality of pseudo reference systems can be addressed by a probabilistic learning model. In particular, we use regression because its problem formulation fits naturally with the objective of MT evaluations. In regression learning, we are interested in approximating 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, $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$, is minimized according to a loss function.

In the context of MT evaluation, y is the "true" quantitative measure of translation quality for an input sentence[1]. The function f represents a mathematical model of human judgments of translations; an input sentence is represented as a feature vector, $\mathbf{x}$, which contains the information that can be extracted from the input sentence (possibly including comparisons against some reference sentences) that are relevant to computing y. Determining the set of relevant features for this modeling is on-going re-

[1] Perhaps even more so than grammaticality judgments, there is variability in people's judgments of translation quality. However, like grammaticality judgments, people do share some similarities in their judgments at a coarse-grained level. Ideally, what we refer to as the true value of translational quality should reflect the consensus judgments of all people.

search. In this work, we consider some of the more widely used metrics as features. Our full feature vector consists of $r \times 18$ adequacy features, where r is the number of reference systems used, and 26 fluency features:

Adequacy features: These include features derived from BLEU (e.g., n-gram precision, where $1 \leq n \leq 5$, length ratios), PER, WER, features derived from METEOR (precision, recall, fragmentation), and ROUGE-related features (non-consecutive bigrams with a gap size of g, where $1 \leq g \leq 5$ and longest common subsequence).

Fluency features: We consider both string-level features such as computing n-gram precision against a target-language corpus as well as several syntax-based features. We parse each input sentence into a dependency tree and compared aspects of it against a large target-language dependency treebank. In addition to adapting the idea of Head Word Chains (Liu and Gildea, 2005), we also compared the input sentence's argument structures against the treebank for certain syntactic categories.

Due to the large feature space to explore, we chose to work with support vector regression as the learning algorithm. As its loss function, support vector regression uses an ϵ-insensitive error function, which allows for errors within a margin of a small positive value, ϵ, to be considered as having zero error (cf. Bishop (2006), pp.339-344). Like its classification counterpart, this is a kernel-based algorithm that finds sparse solutions so that scores for new test instances are efficiently computed based on a subset of the most informative training examples. In this work, Gaussian kernels are used.

The cost of regression learning is that it requires training examples that are manually assessed by human judges. However, compared to the cost of creating new references whenever new (test) sentences are evaluated, the effort of creating human assessment training data is a limited (ideally, one-time) cost. Moreover, there is already a sizable collection of human assessed data for a range of MT systems through multiple years of the NIST MT Eval efforts. Our experiments suggest that there is enough assessed data to train the proposed regression model.

Aside from reducing the cost of developing human reference translations, the proposed metric also provides an alternative perspective on automatic MT evaluation that may be informative in its own right. We conjecture that a metric that compares inputs against a diverse population of differently imperfect sentences may be more discriminative in judging translation systems than solely comparing against gold standards. That is, two sentences may be considered equally bad from the perspective of a gold standard, but subtle differences between them may become more prominent if they are compared against sentences in their peer group.

4 Experiments

We conducted experiments to determine the feasibility of the proposed approach and to address the following questions: (1) How informative are pseudo references in-and-of themselves? Does varying the number and/or the quality of the references have an impact on the metrics? (2) What are the contributions of the adequacy features versus the fluency features to the learning-based metric? (3) How do the quality and distribution of the training examples, together with the quality of the pseudo references, impact the metric training? (4) Do these factors impact the metric's ability in assessing sentences produced within a single MT system? How does that system's quality affect metric performance?

4.1 Data preparation and Experimental Setup

The implementation of support vector regression used for these experiments is SVM-Light (Joachims, 1999). We performed all experiments using the 2004 NIST Chinese MT Eval dataset. It consists of 447 source sentences that were translated by four human translators as well as ten MT systems. Each machine translated sentence was evaluated by two human judges for their fluency and adequacy on a 5-point scale[2]. To remove the bias in the distributions of scores between different judges, we follow the normalization procedure described by Blatz et al. (2003). The two judge's total scores (i.e., sum of the normalized fluency and adequacy scores) are then averaged.

[2]The NIST human judges use human reference translations when making assessments; however, our approach is generally applicable when the judges are bilingual speakers who compare source sentences with translation outputs.

We chose to work with this NIST dataset because it contains numerous systems that span over a range of performance levels (see Table 1 for a ranking of the systems and their averaged human assessment scores). This allows us to have control over the variability of the experiments while answering the questions we posed above (such as the quality of the systems providing the pseudo references, the quality of MT systems being evaluated, and the diversity over the distribution of training examples).

Specifically, we reserved four systems (MT2, MT5, MT6, and MT9) for the role of pseudo references. Sentences produced by the remaining six systems are used as evaluative data. This set includes the best and worst systems so that we can see how well the metrics performs on sentences that are better (or worse) than the pseudo references. Metrics that require no learning are directly applied onto all sentences of the evaluative set. For the learning-based metrics, we perform six-fold cross validation on the evaluative dataset. Each fold consists of sentences from one MT system. In a round robin fashion, each fold serves as the test set while the other five are used for training and heldout. Thus, the trained models have seen neither the test instances nor other instances from the MT system that produced them.

A metric is evaluated based on its Spearman rank correlation coefficient between the scores it gave to the evaluative dataset and human assessments for the same data. The correlation coefficient is a real number between -1, indicating perfect negative correlations, and +1, indicating perfect positive correlations. To compare the relative quality of different metrics, we apply bootstrapping re-sampling on the data, and then use paired t-test to determine the statistical significance of the correlation differences (Koehn, 2004). For the results we report, unless explicitly mentioned, all stated comparisons are statistically significant with 99.8% confidence. We include two standard reference-based metrics, BLEU and METEOR, as baseline comparisons. BLEU is smoothed (Lin and Och, 2004b), and it considers only matching up to bigrams because this has higher correlations with human judgments than when higher-ordered n-grams are included.

SysID	Human-assessment score
MT1	0.661
MT2	**0.626**
MT3	0.586
MT4	0.578
MT5	**0.537**
MT6	**0.530**
MT7	0.530
MT8	0.375
MT9	**0.332**
MT10	0.243

Table 1: The human-judged quality of ten participating systems in the NIST 2004 Chinese MT Evaluation. We used four systems as references (highlighted in boldface) and the data from the remaining six for training and evaluation.

4.2 Pseudo Reference Variations vs. Metrics

We first compare different metrics' performance on the six-system evaluative dataset under different configurations of human and/or pseudo references. For the case when only one human reference is used, the reference was chosen at random from the 2004 NIST Eval dataset[3]. The correlation results on the evaluative dataset are summarized in Table 2.

Some trends are as expected: comparing within a metric, having four references is better than having just one; having human references is better than an equal number of system references; having a high quality system as reference is better than one with low quality. Perhaps more surprising is the consistent trend that metrics do significantly better with four MT references than with one human reference, and they do almost as well as using four human references. The results show that pseudo references are informative, as standard metrics were able to make use of the pseudo references and achieve higher correlations than judging from fluency alone. However, higher correlations are achieved when learning with regression, suggesting that the trained metrics are better at interpreting comparisons against pseudo references.

Comparing within each reference configuration, the regression-trained metric that includes both ad-

[3]One reviewer asked about the quality this human's translations. Although we were not given official rankings of the human references, we compared each person against the other three using MT evaluation metrics and found this particular translator to rank third, though the quality of all four are significantly higher than even the best MT systems.

equacy and fluency features always has the highest correlations. If the metric consists of only adequacy features, its performance degrades with the decreasing quality of the references. At another extreme, a metric based only on fluency features has an overall correlation rate of 0.459, which is lower than most correlations reported in Table 2. This confirms the importance of modeling adequacy; even a single mid-quality MT system may be an informative pseudo reference. Finally, we note that a regression-trained metric with the full features set that compares against 4 pseudo references has a higher correlation than BLEU with four human references. These results suggest that the feedback from the human assessed training examples was able to help the learning algorithm to combine different features to form a better composite metric.

4.3 Sentence-Level Evaluation on Single Systems

To explore the interaction between the quality of the reference MT systems and that of the test MT systems, we further study the following pseudo reference configurations: all four systems, a high-quality system with a medium quality system, two systems of medium-quality, one medium with one poor system, and only the high-quality system. For each pseudo reference configuration, we consider three metrics: BLEU, METEOR, and the regression-trained metric (using the full feature set). Each metric evaluates sentences from four test systems of varying quality: the best system in the dataset (MT1), the worst in the set (MT10), and two mid-ranged systems (MT4 and MT7). The correlation coefficients are summarized in Table 3. Each row specifies a metric/reference-type combination; each column specifies an MT system being evaluated (using sentences from all other systems as training examples). The fluency-only metric and standard metrics using four human references are baselines.

The overall trends at the dataset level generally also hold for the per-system comparisons. With the exception of the evaluation of MT10, regression-based metrics always has higher correlations than standard metrics that use the same reference configuration (comparing correlation coefficients within each cell). When the best MT reference system (MT2) is included as pseudo references, regression-based metrics are typically better than or not statistically different from standard applications of BLEU and METEOR with 4 human references. Using the two mid-quality MT systems as references (MT5 and MT6), regression metrics yield correlations that are only slightly lower than standard metrics with human references. These results support our conjecture that comparing against multiple systems is informative.

The poorer performances of the regression-based metrics on MT10 point out an asymmetry in the learning approach. The regression model aims to learn a function that approximates human judgments of translated sentences through training examples. In the space of all possible MT outputs, the neighborhood of good translations is much smaller than that of bad translations. Thus, as long as the regression models sees some examples of sentences with high assessment scores during training, it should have a much better estimation of the characteristics of good translations. This idea is supported by the experimental data. Consider the scenario of evaluating MT1 while using two mid-quality MT systems as references. Although the reference systems are not as high quality as the system under evaluation, and although the training examples shown to the regression model were also generated by systems whose overall quality was rated lower, the trained metric was reasonably good at ranking sentences produced by MT1. In contrast, the task of evaluating sentences from MT10 is more difficult for the learning approach, perhaps because it is sufficiently different from all training and reference systems. Correlations might be improved with additional reference systems.

4.4 Discussions

The design of these experiments aims to simulate practical situations to use our proposed metrics. For the more frequently encountered language pairs, it should be possible to find at least two mid-quality (or better) MT systems to serve as pseudo references. For example, one might use commercial off-the-shelf systems, some of which are free over the web. For less commonly used languages, one might use open source research systems (Al-Onaizan et al., 1999; Burbank et al., 2005).

Datasets from formal evaluation events such as

Ref type and #	Ref Sys.	BLEU-S(2)	METEOR	Regr (adq. only)	Regr (full)
4 Humans	all humans	0.628	0.591	0.588	**0.644**
1 Human	HRef #3	0.536	0.512	0.487	0.597
4 Systems	all MTRefs	0.614	0.583	0.584	*0.632*
2 Systems	Best 2 MTRefs	0.603	0.577	0.573	0.620
	Mid 2 MTRefs	0.579	0.555	0.528	0.608
	Worst 2 MTRefs	0.541	0.508	0.467	0.581
1 System	Best MTRef	0.576	0.559	0.534	0.596
	Mid MTRef (MT5)	0.538	0.528	0.474	0.577
	Worst MTRef	0.371	0.329	0.151	0.495

Table 2: Comparisons of metrics (columns) using different types of references (rows). The full regression-trained metric has the highest correlation (shown in boldface) when four human references are used; it has the second highest correlation rate (shown in italic) when four MT system references are used instead. A regression-trained metric with only fluency features has a correlation coefficient of 0.459.

Ref Type	Metric	MT-1	MT-4	MT-7	MT-10
No ref	Regr.	0.367	0.316	0.301	-0.045
4 human refs	Regr.	**0.538***	**0.473***	**0.459***	0.247
	BLEU-S(2)	0.466	0.419	0.397	0.321*
	METEOR	0.464	0.418	0.410	0.312
4 MTRefs	Regr.	**0.498**	**0.429**	**0.421**	0.243
	BLEU-S(2)	0.386	0.349	0.404	0.240
	METEOR	0.445	0.354	0.333	0.243
Best 2 MTRefs	Regr.	**0.492**	*0.418*	0.403	0.201
	BLEU-S(2)	0.391	0.330	0.394	0.268
	METEOR	0.430	0.333	0.327	0.267
Mid 2 MTRefs	Regr.	0.450	0.413	0.388	0.219
	BLEU-S(2)	0.362	0.314	0.310	0.282
	METEOR	0.391	0.315	0.284	0.274
Worst 2 MTRefs	Regr.	0.430	0.386	0.365	0.158
	BLEU-S(2)	0.320	0.298	0.316	0.223
	METEOR	0.351	0.306	0.302	0.228
Best MTRef	Regr.	0.461	0.401	**0.414**	0.122
	BLEU-S(2)	0.371	0.330	0.380	0.242
	METEOR	0.375	0.318	0.392	0.283

Table 3: Correlation comparisons of metrics by test systems. For each test system (columns) the overall highest correlations is distinguished by an asterisk (*); correlations higher than *standard metrics using human-references* are highlighted in boldface; those that are statistically comparable to them are italicized.

NIST MT Evals, which contains human assessed MT outputs for a variety of systems, can be used for training examples. Alternatively, one might directly recruit human judges to assess sample sentences from the system(s) to be evaluated. This should result in better correlations than what we reported here, since the human assessed training examples will be more similar to the test instances than the setup in our experiments.

In developing new MT systems, pseudo references may supplement the single human reference translations that could be extracted from a parallel text. Using the same setup as Exp. 1 (see Table 2), adding pseudo references does improve correlations. Adding four pseudo references to the single human reference raises the correlation coefficient to 0.650 (from 0.597) for the regression metric. Adding them to four human references results in a correlation coefficient of 0.660 (from 0.644)[4].

5 Conclusion

In this paper, we have presented a method for developing sentence-level MT evaluation metrics without using human references. We showed that by learning from human assessed training examples,

[4]BLEU with four human references has a correlation of 0.628. Adding four pseudo references increases BLEU to 0.650.

the regression-trained metric can evaluate an input sentence by comparing it against multiple machine-generated pseudo references and other target language resources. Our experimental results suggest that the resulting metrics are robust even when the sentences under evaluation are from a system of higher quality than the systems serving as references. We observe that regression metrics that use multiple pseudo references often have comparable or higher correlation rates with human judgments than standard reference-based metrics. Our study suggests that in conjunction with regression training, multiple imperfect references may be as informative as gold-standard references.

Acknowledgments

This work has been supported by NSF Grants IIS-0612791 and IIS-0710695. We would like to thank Ric Crabbe, Dan Gildea, Alon Lavie, Stuart Shieber, and Noah Smith and the anonymous reviewers for their suggestions. We are also grateful to NIST for making their assessment data available to us.

References

Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, I. Dan Melamed, Franz-Josef Och, David Purdy, Noah A. Smith, and David Yarowsky. 1999. Statistical machine translation. Technical report, JHU. citeseer.nj.nec.com/al-onaizan99statistical.html.

Joshua S. Albrecht and Rebecca Hwa. 2007. A re-examination of machine learning approaches for sentence-level MT evaluation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-2007)*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for MT evaluation with improved correlation with human judgments. In *ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, June.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer Verlag.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2003. Confidence estimation for machine translation. Technical Report Natural Language Engineering Workshop Final Report, Johns Hopkins University.

Andrea Burbank, Marine Carpuat, Stephen Clark, Markus Dreyer, Declan Groves Pamela. Fox, Keith Hall, Mary Hearne, I. Dan Melamed, Yihai Shen, Andy Way, Ben Wellington, and Dekai Wu. 2005. Final report of the 2005 language engineering workshop on statistical machine translation by parsing. Technical Report Natural Language Engineering Workshop Final Report, "JHU".

Simon Corston-Oliver, Michael Gamon, and Chris Brockett. 2001. A machine learning approach to the automatic evaluation of machine translation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, July.

Michael Gamon, Anthony Aue, and Martine Smets. 2005. Sentence-level MT evaluation without reference translations: Beyond language modeling. In *European Association for Machine Translation (EAMT)*, May.

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.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP-04)*.

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)*, Baltimore, MD, October.

Chin-Yew Lin and Franz Josef Och. 2004a. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, July.

Chin-Yew Lin and Franz Josef Och. 2004b. Orange: a method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, August.

Ding Liu and Daniel Gildea. 2005. Syntactic features for evaluation of machine translation. In *ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, June.

Ding Liu and Daniel Gildea. 2007. Source-language features and maximum correlation training for machine translation evaluation. In *Proceedings of the HLT/NAACL-2007*, April.

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*, Philadelphia, PA.

Christopher Quirk. 2004. Training a sentence-level machine translation confidence measure. In *Proceedings of LREC 2004*.

Nicola Ueffing, Klaus Macherey, and Hermann Ney. 2003. Confidence measures for statistical machine translation. In *Machine Translation Summit IX*, pages 394–401, September.

Bootstrapping Word Alignment via Word Packing

Yanjun Ma, Nicolas Stroppa, Andy Way
School of Computing
Dublin City University
Glasnevin, Dublin 9, Ireland
{yma,nstroppa,away}@computing.dcu.ie

Abstract

We introduce a simple method to pack words for statistical word alignment. Our goal is to simplify the task of automatic word alignment by packing several consecutive words together when we believe they correspond to a single word in the opposite language. This is done using the word aligner itself, i.e. by bootstrapping on its output. We evaluate the performance of our approach on a Chinese-to-English machine translation task, and report a 12.2% relative increase in BLEU score over a state-of-the art phrase-based SMT system.

1 Introduction

Automatic word alignment can be defined as the problem of determining a translational correspondence at word level given a parallel corpus of aligned sentences. Most current statistical models (Brown et al., 1993; Vogel et al., 1996; Deng and Byrne, 2005) treat the aligned sentences in the corpus as sequences of tokens that are meant to be words; the goal of the alignment process is to find links between source and target words. Before applying such aligners, we thus need to segment the sentences into words – a task which can be quite hard for languages such as Chinese for which word boundaries are not orthographically marked. More importantly, however, this segmentation is often performed in a *monolingual* context, which makes the word alignment task more difficult since different languages may realize the same concept using varying numbers of words (see e.g. (Wu, 1997)). Moreover, a segmentation considered to be "good" from a monolingual point of view may be unadapted for training alignment models.

Although some statistical alignment models allow for 1-to-n word alignments for those reasons, they rarely question the monolingual tokenization and the basic unit of the alignment process remains the word. In this paper, we focus on 1-to-n alignments with the goal of simplifying the task of automatic word aligners by *packing* several consecutive words together when we believe they correspond to a single word in the opposite language; by identifying enough such cases, we reduce the number of 1-to-n alignments, thus making the task of word alignment both easier and more natural.

Our approach consists of using the output from an existing statistical word aligner to obtain a set of candidates for word packing. We evaluate the reliability of these candidates, using simple metrics based on co-occurence frequencies, similar to those used in associative approaches to word alignment (Kitamura and Matsumoto, 1996; Melamed, 2000; Tiedemann, 2003). We then modify the segmentation of the sentences in the parallel corpus according to this packing of words; these modified sentences are then given back to the word aligner, which produces new alignments. We evaluate the validity of our approach by measuring the influence of the alignment process on a Chinese-to-English Machine Translation (MT) task.

The remainder of this paper is organized as follows. In Section 2, we study the case of 1-to-n word alignment. Section 3 introduces an automatic method to pack together groups of consecutive

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 304–311,
Prague, Czech Republic, June 2007.

		1:0	1:1	1:2	1:3	1:n ($n > 3$)
IWSLT	Chinese–English	21.64	63.76	9.49	3.36	1.75
IWSLT	English–Chinese	29.77	57.47	10.03	1.65	1.08
IWSLT	Italian–English	13.71	72.87	9.77	3.23	0.42
IWSLT	English–Italian	20.45	71.08	7.02	0.9	0.55
Europarl	Dutch–English	24.71	67.04	5.35	1.4	1.5
Europarl	English–Dutch	23.76	69.07	4.85	1.2	1.12

Table 1: Distribution of alignment types for different language pairs (%)

words based on the output from a word aligner. In Section 4, the experimental setting is described. In Section 5, we evaluate the influence of our method on the alignment process on a Chinese to English MT task, and experimental results are presented. Section 6 concludes the paper and gives avenues for future work.

2 The Case of 1-to-n Alignment

The same concept can be expressed in different languages using varying numbers of words; for example, a single Chinese word may surface as a compound or a collocation in English. This is frequent for languages as different as Chinese and English. To quickly (and approximately) evaluate this phenomenon, we trained the statistical IBM word-alignment model 4 (Brown et al., 1993),[1] using the GIZA++ software (Och and Ney, 2003) for the following language pairs: Chinese–English, Italian–English, and Dutch–English, using the IWSLT-2006 corpus (Takezawa et al., 2002; Paul, 2006) for the first two language pairs, and the Europarl corpus (Koehn, 2005) for the last one. These asymmetric models produce 1-to-n alignments, with $n \geq 0$, in both directions. Here, it is important to mention that the segmentation of sentences is performed totally independently of the bilingual alignment process, i.e. it is done in a *monolingual* context. For European languages, we apply the maximum-entropy based tokenizer of OpenNLP[2]; the Chinese sentences were human segmented (Paul, 2006).

In Table 1, we report the frequencies of the different types of alignments for the various languages and directions. As expected, the number of 1:n alignments with $n \neq 1$ is high for Chinese–English ($\simeq 40\%$), and significantly higher than for the European languages. The case of 1-to-n alignments is, therefore, obviously an important issue when dealing with Chinese–English word alignment.[3]

2.1 The Treatment of 1-to-n Alignments

Fertility-based models such as IBM models 3, 4, and 5 allow for alignments between one word and several words (1-to-n or 1:n alignments in what follows), in particular for the reasons specified above. They can be seen as extensions of the simpler IBM models 1 and 2 (Brown et al., 1993). Similarly, Deng and Byrne (2005) propose an HMM framework capable of dealing with 1-to-n alignment, which is an extension of the original model of (Vogel et al., 1996).

However, these models rarely question the monolingual tokenization, i.e. the basic unit of the alignment process is the word.[4] One alternative to extending the expressivity of one model (and usually its complexity) is to focus on the *input representation*; in particular, we argue that the alignment process can benefit from a simplification of the input, which consists of trying to reduce the number of 1-to-n alignments to consider. Note that the need to consider segmentation and alignment at the same time is also mentioned in (Tiedemann, 2003), and related issues are reported in (Wu, 1997).

2.2 Notation

While in this paper, we focus on Chinese–English, the method proposed is applicable to any language

[1] More specifically, we performed 5 iterations of Model 1, 5 iterations of HMM, 5 iterations of Model 3, and 5 iterations of Model 4.

[2] `http://opennlp.sourceforge.net/`.

[3] Note that a 1:0 alignment may denote a failure to capture a 1:n alignment with $n > 1$.

[4] Interestingly, this is actually even the case for approaches that directly model alignments between phrases (Marcu and Wong, 2002; Birch et al., 2006).

pair – even for closely related languages, we expect improvements to be seen. The notation however assume Chinese–English MT. Given a Chinese sentence c_1^J consisting of J words $\{c_1, \ldots, c_J\}$ and an English sentence e_1^I consisting of I words $\{e_1, \ldots, e_I\}$, $A_{C \to E}$ (resp. $A_{E \to C}$) will denote a Chinese-to-English (resp. an English-to-Chinese) word alignment between c_1^J and e_1^I. Since we are primarily interested in 1-to-n alignments, $A_{C \to E}$ can be represented as a set of pairs $a_j = \langle c_j, E_j \rangle$ denoting a link between one single Chinese word c_j and a few English words E_j (and similarly for $A_{E \to C}$). The set E_j is empty if the word c_j is not aligned to any word in e_1^I.

3 Automatic Word Repacking

Our approach consists of packing consecutive words together when we believe they correspond to a single word in the other language. This bilingually motivated packing of words changes the basic unit of the alignment process, and simplifies the task of automatic word alignment. We thus minimize the number of 1-to-n alignments in order to obtain more comparable segmentations in the two languages. In this section, we present an automatic method that builds upon the output from an existing automatic word aligner. More specifically, we (i) use a word aligner to obtain 1-to-n alignments, (ii) extract candidates for word packing, (iii) estimate the reliability of these candidates, (iv) replace the groups of words to pack by a single token in the parallel corpus, and (v) re-iterate the alignment process using the updated corpus. The first three steps are performed in both directions, and produce two *bilingual dictionaries* (source-target and target-source) of groups of words to pack.

3.1 Candidate Extraction

In the following, we assume the availability of an automatic word aligner that can output alignments $A_{C \to E}$ and $A_{E \to C}$ for any sentence pair (c_1^J, e_1^I) in a parallel corpus. We also assume that $A_{C \to E}$ and $A_{E \to C}$ contain $1{:}\,n$ alignments. Our method for repacking words is very simple: whenever a single word is aligned with several consecutive words, they are considered candidates for repacking. Formally, given an alignment $A_{C \to E}$ between c_1^J and e_1^I, if $a_j = \langle c_j, E_j \rangle \in A_{C \to E}$, with $E_j = \{e_{j_1}, \ldots, e_{j_m}\}$ and $\forall k \in [\![1, m-1]\!]$, $j_{k+1} - j_k = 1$, then the alignment a_j between c_j and the sequence of words E_j is considered a candidate for word repacking. The same goes for $A_{E \to C}$. Some examples of such 1-to-n alignments between Chinese and English (in both directions) we can derive automatically are displayed in Figure 1.

白葡萄酒: white wine	closest: 最 近
百货公司: department store	fifteen: 十 五
抱歉: excuse me	fine: 很 好
报警: call the police	flight: 次 航班
杯: cup of	get: 拿 到
必须: have to	here: 在 这里

Figure 1: Example of 1-to-n word alignments between Chinese and English

3.2 Candidate Reliability Estimation

Of course, the process described above is error-prone and if we want to change the input to give to the word aligner, we need to make sure that we are not making harmful modifications.[5] We thus additionally evaluate the reliability of the candidates we extract and filter them before inclusion in our bilingual dictionary. To perform this filtering, we use two simple statistical measures. In the following, $a_j = \langle c_j, E_j \rangle$ denotes a candidate.

The first measure we consider is co-occurrence frequency ($COOC(c_j, E_j)$), i.e. the number of times c_j and E_j co-occur in the bilingual corpus. This very simple measure is frequently used in associative approaches (Melamed, 1997; Tiedemann, 2003). The second measure is the alignment confidence, defined as

$$AC(a_j) = \frac{C(a_j)}{COOC(c_j, E_j)},$$

where $C(a_j)$ denotes the number of alignments proposed by the word aligner that are identical to a_j. In other words, $AC(a_j)$ measures how often the

[5] Consequently, if we compare our approach to the problem of collocation identification, we may say that we are more interested in precision than recall (Smadja et al., 1996). However, note that our goal is not recognizing specific sequences of words such as compounds or collocations; it is making (bilingually motivated) changes that simplify the alignment process.

aligner aligns c_j and E_j when they co-occur. We also impose that $|E_j| \leq k$, where k is a fixed integer that may depend on the language pair (between 3 and 5 in practice). The rationale behind this is that it is very rare to get reliable alignment between one word and k consecutive words when k is high.

The candidates are included in our bilingual dictionary if and only if their measures are above some fixed thresholds t_{cooc} and t_{ac}, which allow for the control of the size of the dictionary and the quality of its contents. Some other measures (including the Dice coefficient) could be considered; however, it has to be noted that we are more interested here in the filtering than in the discovery of alignment, since our method builds upon an existing aligner. Moreover, we will see that even these simple measures can lead to an improvement of the alignment process in a MT context (cf. Section 5).

3.3 Bootstrapped Word Repacking

Once the candidates are extracted, we repack the words in the bilingual dictionaries constructed using the method described above; this provides us with an updated training corpus, in which some word sequences have been replaced by a single token. This update is totally naive: if an entry $a_j = \langle c_j, E_j \rangle$ is present in the dictionary and matches one sentence pair (c_1^J, e_1^I) (i.e. c_j and E_j are respectively contained in c_1^J and e_1^I), then we replace the sequence of words E_j with a single token which becomes a new lexical unit.[6] Note that this replacement occurs even if no alignment was found between c_j and E_j for the pair (c_1^J, e_1^I). This is motivated by the fact that the filtering described above is quite conservative; we trust the entry a_i to be correct. This update is performed in both directions. It is then possible to run the word aligner using the updated (simplified) parallel corpus, in order to get new alignments. By performing a deterministic word packing, we avoid the computation of the fertility parameters associated with fertility-based models.

Word packing can be applied several times: once we have grouped some words together, they become the new basic unit to consider, and we can re-run the same method to get additional groupings. However, we have not seen in practice much benefit from running it more than twice (few new candidates are extracted after two iterations).

It is also important to note that this process is bilingually motivated and strongly depends on the language pair. For example, *white wine*, *excuse me*, *call the police*, and *cup of* (cf. Figure 1) translate respectively as *vin blanc*, *excusez-moi*, *appellez la police*, and *tasse de* in French. Those groupings would not be found for a language pair such as French–English, which is consistent with the fact that they are less useful for French–English than for Chinese–English in a MT perspective.

3.4 Using Manually Developed Dictionaries

We wanted to compare this automatic approach to manually developed resources. For this purpose, we used a dictionary built by the MT group of Harbin Institute of Technology, as a preprocessing step to Chinese–English word alignment, and motivated by several years of Chinese–English MT practice. Some examples extracted from this resource are displayed in Figure 2.

有: there is
想要: want to
不必: need not
前面: in front of
一: as soon as
看: look at

Figure 2: Examples of entries from the manually developed dictionary

4 Experimental Setting

4.1 Evaluation

The intrinsic quality of word alignment can be assessed using the Alignment Error Rate (AER) metric (Och and Ney, 2003), that compares a system's alignment output to a set of gold-standard alignment. While this method gives a direct evaluation of the quality of word alignment, it is faced with several limitations. First, it is really difficult to build a reliable and objective gold-standard set, especially for languages as different as Chinese and English. Second, an increase in AER does not necessarily imply an improvement in translation quality (Liang et al., 2006) and vice-versa (Vilar et al., 2006). The

[6] In case of overlap between several groups of words to replace, we select the one with highest confidence (according to t_{ac}).

relationship between word alignments and their impact on MT is also investigated in (Ayan and Dorr, 2006; Lopez and Resnik, 2006; Fraser and Marcu, 2006). Consequently, we chose to extrinsically evaluate the performance of our approach via the translation task, i.e. we measure the influence of the alignment process on the final translation output. The quality of the translation output is evaluated using BLEU (Papineni et al., 2002).

4.2 Data

The experiments were carried out using the Chinese–English datasets provided within the IWSLT 2006 evaluation campaign (Paul, 2006), extracted from the Basic Travel Expression Corpus (BTEC) (Takezawa et al., 2002). This multilingual speech corpus contains sentences similar to those that are usually found in phrase-books for tourists going abroad. Training was performed using the default training set, to which we added the sets devset1, devset2, and devset3.[7] The English side of the test set was not available at the time we conducted our experiments, so we split the development set (devset 4) into two parts: one was kept for testing (200 aligned sentences) with the rest (289 aligned sentences) used for development purposes.

As a pre-processing step, the English sentences were tokenized using the maximum-entropy based tokenizer of the OpenNLP toolkit, and case information was removed. For Chinese, the data provided were tokenized according to the output format of ASR systems, and human-corrected (Paul, 2006). Since segmentations are human-corrected, we are sure that they are good from a monolingual point of view. Table 2 contains the various corpus statistics.

4.3 Baseline

We use a standard log-linear phrase-based statistical machine translation system as a baseline: GIZA++ implementation of IBM word alignment model 4 (Brown et al., 1993; Och and Ney, 2003),[8] the refinement and phrase-extraction heuristics described in (Koehn et al., 2003), minimum-error-rate training (Och, 2003) using Phramer (Olteanu et al., 2006), a 3-gram language model with Kneser-Ney smoothing trained with SRILM (Stolcke, 2002) on the English side of the training data and Pharaoh (Koehn, 2004) with default settings to decode. The log-linear model is also based on standard features: conditional probabilities and lexical smoothing of phrases in both directions, and phrase penalty (Zens and Ney, 2004).

		Chinese	English
Train	Sentences	41,465	
	Running words	361,780	375,938
	Vocabulary size	11,427	9,851
Dev.	Sentences	289 (7 refs.)	
	Running words	3,350	26,223
	Vocabulary size	897	1,331
Eval.	Sentences	200 (7 refs.)	
	Running words	1,864	14,437
	Vocabulary size	569	1,081

Table 2: Chinese–English corpus statistics

5 Experimental Results

5.1 Results

The initial word alignments are obtained using the baseline configuration described above. From these, we build two bilingual 1-to-n dictionaries (one for each direction), and the training corpus is updated by repacking the words in the dictionaries, using the method presented in Section 2. As previously mentioned, this process can be repeated several times; at each step, we can also choose to exploit only one of the two available dictionaries, if so desired. We then extract aligned phrases using the same procedure as for the baseline system; the only difference is the basic unit we are considering. Once the phrases are extracted, we perform the estimation of the features of the log-linear model and unpack the grouped words to recover the initial words. Finally, minimum-error-rate training and decoding are performed.

The various parameters of the method (k, t_{cooc}, t_{ac}, cf. Section 2) have been optimized on the development set. We found out that it was enough to perform two iterations of repacking: the optimal set of values was found to be $k = 3$, $t_{ac} = 0.5$, $t_{cooc} = 20$ for the first iteration, and $t_{cooc} = 10$ for the second

[7] More specifically, we choose the first English reference from the 7 references and the Chinese sentence to construct new sentence pairs.

[8] Training is performed using the same number of iterations as in Section 2.

	BLEU[%]
Baseline	15.14
n=1. with C-E dict.	15.92
n=1. with E-C dict.	15.77
n=1. with both	**16.59**
n=2. with C-E dict.	**16.99**
n=2. with E-C dict.	16.59
n=2. with both	16.88

Table 3: Influence of word repacking on Chinese-to-English MT

iteration, for both directions.[9] In Table 3, we report the results obtained on the test set, where n denotes the iteration. We first considered the inclusion of only the Chinese–English dictionary, then only the English–Chinese dictionary, and then both.

After the first step, we can already see an improvement over the baseline when considering one of the two dictionaries. When using both, we observe an increase of 1.45 BLEU points, which corresponds to a 9.6% relative increase. Moreover, we can gain from performing another step. However, the inclusion of the English–Chinese dictionary is harmful in this case, probably because 1-to-n alignments are less frequent for this direction, and have been captured during the first step. By including the Chinese–English dictionary only, we can achieve an increase of 1.85 absolute BLEU points (12.2% relative) over the initial baseline.[10]

Quality of the Dictionaries To assess the quality of the extraction procedure, we simply manually evaluated the ratio of incorrect entries in the dictionaries. After one step of word packing, the Chinese–English and the English–Chinese dictionaries respectively contain 7.4% and 13.5% incorrect entries. After two steps of packing, they only contain 5.9% and 10.3% incorrect entries.

5.2 Alignment Types

Intuitively, the word alignments obtained after word packing are more likely to be 1-to-1 than before. Indeed, the word sequences in one language that usually align to one single word in the other language have been grouped together to form one single token. Table 4 shows the detail of the distribution of alignment types after one and two steps of automatic repacking. In particular, we can observe that the $1\!:\!1$ alignments are more frequent after the application of repacking: the ratio of this type of alignment has increased by 7.81% for Chinese–English and 5.26% for English–Chinese.

		$1\!:\!0$	$1\!:\!1$	$1\!:\!2$	$1\!:\!3$	$1\!:\!n$ $(n>3)$
C-E	Base.	21.64	**63.76**	9.49	3.36	1.75
	n=1	19.69	69.43	6.32	2.79	1.78
	n=2	19.67	**71.57**	4.87	2.12	1.76
E-C	Base.	29.77	**57.47**	10.03	1.65	1.08
	n=1	26.59	61.95	8.82	1.55	1.09
	n=2	25.10	**62.73**	9.38	1.68	1.12

Table 4: Distribution of alignment types ($\%$)

5.3 Influence of Word Segmentation

To test the influence of the initial word segmentation on the process of word packing, we considered an additional segmentation configuration, based on an automatic segmenter combining rule-based and statistical techniques (Zhao et al., 2001).

	BLEU[%]
Original segmentation	15.14
Original segmentation + Word packing	**16.99**
Automatic segmentation	14.91
Automatic segmentation + Word packing	**17.51**

Table 5: Influence of Chinese segmentation

The results obtained are displayed in Table 5. As expected, the automatic segmenter leads to slightly lower results than the human-corrected segmentation. However, the proposed method seems to be beneficial irrespective of the choice of segmentation. Indeed, we can also observe an improvement in the new setting: 2.6 points absolute increase in BLEU (17.4% relative).[11]

[9] The parameters k, t_{ac}, and t_{cooc} are optimized for each step, and the alignment obtained using the best set of parameters for a given step are used as input for the following step.

[10] Note that this setting (using both dictionaries for the first step and only the Chinese dictionary for the second step) is also the best setting on the development set.

[11] We could actually consider an extreme case, which would consist of splitting the sentences into characters, i.e. each character would be blindly treated as one word. The segmentation

5.4 Exploiting Manually Developed Resources

We also compared our technique for automatic packing of words with the exploitation of manually developed resources. More specifically, we used a 1-to-n Chinese–English bilingual dictionary, described in Section 3.4, and used it in place of the automatically acquired dictionary. Words are thus grouped according to this dictionary, and we then apply the same word aligner as for previous experiments. In this case, since we are not bootstrapping from the output of a word aligner, this can actually be seen as a pre-processing step prior to alignment. These resources follow more or less the same format as the output of the word segmenter mentioned in Section 5.1.2 (Zhao et al., 2001), so the experiments are carried out using this segmentation.

	BLEU[%]
Baseline	14.91
Automatic word packing	**17.51**
Packing with "manual" dictionary	16.15

Table 6: Exploiting manually developed resources

The results obtained are displayed in Table 6.We can observe that the use of the manually developed dictionary provides us with an improvement in translation quality: 1.24 BLEU points absolute (8.3% relative). However, there does not seem to be a clear gain when compared with the automatic method. Even if those manual resources were extended, we do not believe the improvement is sufficient enough to justify this additional effort.

6 Conclusion and Future Work

In this paper, we have introduced a simple yet effective method to pack words together in order to give a different and simplified input to automatic word aligners. We use a bootstrap approach in which we first extract 1-to-n word alignments using an existing word aligner, and then estimate the confidence of those alignments to decide whether or not the n words have to be grouped; if so, this group is considered a new basic unit to consider. We can finally re-apply the word aligner to the updated sentences.

We have evaluated the performance of our approach by measuring the influence of this process on a Chinese-to-English MT task, based on the IWSLT 2006 evaluation campaign. We report a 12.2% relative increase in BLEU score over a standard phrase-based SMT system. We have verified that this process actually reduces the number of $1{:}\,n$ alignments with $n \neq 1$, and that it is rather independent from the (Chinese) segmentation strategy.

As for future work, we first plan to consider different confidence measures for the filtering of the alignment candidates. We also want to bootstrap on different word aligners; in particular, one possibility is to use the flexible HMM word-to-phrase model of Deng and Byrne (2005) in place of IBM model 4. Finally, we would like to apply this method to other corpora and language pairs.

Acknowledgment

This work is supported by Science Foundation Ireland (grant number OS/IN/1732). Prof. Tiejun Zhao and Dr. Muyun Yang from the MT group of Harbin Institute of Technology, and Yajuan Lv from the Institute of Computing Technology, Chinese Academy of Sciences, are kindly acknowledged for providing us with the Chinese segmenter and the manually developed bilingual dictionary used in our experiments.

References

Necip Fazil Ayan and Bonnie J. Dorr. 2006. Going beyond aer: An extensive analysis of word alignments and their impact on mt. In *Proceedings of COLING-ACL 2006*, pages 9–16, Sydney, Australia.

Alexandra Birch, Chris Callison-Burch, and Miles Osborne. 2006. Constraining the phrase-based, joint probability statistical translation model. In *Proceedings of AMTA 2006*, pages 10–18, Boston, MA.

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

Yonggang Deng and William Byrne. 2005. HMM word and phrase alignment for statistical machine translation. In *Proceedings of HLT-EMNLP 2005*, pages 169–176, Vancouver, Canada.

would thus be completely driven by the *bilingual* alignment process (see also (Wu, 1997; Tiedemann, 2003) for related considerations). In this case, our approach would be similar to the approach of (Xu et al., 2004), except for the estimation of candidates.

Alexander Fraser and Daniel Marcu. 2006. Measuring word alignment quality for statistical machine translation. Technical Report ISI-TR-616, ISI/University of Southern California.

Mihoko Kitamura and Yuji Matsumoto. 1996. Automatic extraction of word sequence correspondences in parallel corpora. In *Proceedings of the 4th Workshop on Very Large Corpora*, pages 79–87, Copenhagen, Denmark.

Philip Koehn, Franz Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003*, pages 48–54, Edmonton, Canada.

Philip Koehn. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proceedings of AMTA 2004*, pages 115–124, Washington, District of Columbia.

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

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of HLT-NAACL 2006*, pages 104–111, New York, NY.

Adam Lopez and Philip Resnik. 2006. Word-based alignment, phrase-based translation: What's the link? In *Proceedings of AMTA 2006*, pages 90–99, Cambridge, MA.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP 2002*, pages 133–139, Morristown, NJ.

I. Dan Melamed. 1997. Automatic discovery of non-compositional compounds in parallel data. In *Proceedings of EMNLP 1997*, pages 97–108, Somerset, New Jersey.

I. Dan Melamed. 2000. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249.

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

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

Marian Olteanu, Chris Davis, Ionut Volosen, and Dan Moldovan. 2006. Phramer - an open source statistical phrase-based translator. In *Proceedings of the NAACL 2006 Workshop on Statistical Machine Translation*, pages 146–149, New York, NY.

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

Michael Paul. 2006. Overview of the IWSLT 2006 Evaluation Campaign. In *Proceedings of IWSLT 2006*, pages 1–15, Kyoto, Japan.

Frank Smadja, Kathleen R. McKeown, and Vasileios Hatzivassiloglou. 1996. Translating collocations for bilingual lexicons: A statistical approach. *Computational Linguistics*, 22(1):1–38.

Andrea Stolcke. 2002. SRILM – An extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, pages 901–904, Denver, Colorado.

T. Takezawa, E. Sumita, F. Sugaya, H. Yamamoto, and S. Yamamoto. 2002. Toward a broad-coverage bilingual corpus for speech translation of travel conversations in the real world. In *Proceedings of LREC 2002*, pages 147–152, Las Palmas, Spain.

Jörg Tiedemann. 2003. Combining clues for word alignment. In *Proceedings of EACL 2003*, pages 339–346, Budapest, Hungary.

David Vilar, Maja Popovic, and Hermann Ney. 2006. AER: Do we need to "improve" our alignments? In *Proceedings of IWSLT 2006*, pages 205–212, Kyoto, Japan.

Stefan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of COLING 1996*, pages 836–841, Copenhagen, Denmark.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Jia Xu, Richard Zens, and Hermann Ney. 2004. Do we need chinese word segmentation for statistical machine translation? In *Proceedings of the Third SIGHAN Workshop on Chinese Language Learning*, pages 122–128, Barcelona, Spain.

Richard Zens and Hermann Ney. 2004. Improvements in phrase-based statistical machine translation. In *Proceedings of HLT-NAACL 2004*, pages 257–264, Boston, MA.

Tiejun Zhao, Yajuan Lü, and Hao Yu. 2001. Increasing accuracy of chinese segmentation with strategy of multi-step processing. *Journal of Chinese Information Processing*, 15(1):13–18.

Improved Word-Level System Combination for Machine Translation

Antti-Veikko I. Rosti and **Spyros Matsoukas** and **Richard Schwartz**
BBN Technologies, 10 Moulton Street
Cambridge, MA 02138
{arosti,smatsouk,schwartz}@bbn.com

Abstract

Recently, confusion network decoding has been applied in machine translation system combination. Due to errors in the hypothesis alignment, decoding may result in ungrammatical combination outputs. This paper describes an improved confusion network based method to combine outputs from multiple MT systems. In this approach, arbitrary features may be added log-linearly into the objective function, thus allowing language model expansion and re-scoring. Also, a novel method to automatically select the hypothesis which other hypotheses are aligned against is proposed. A generic weight tuning algorithm may be used to optimize various automatic evaluation metrics including TER, BLEU and METEOR. The experiments using the 2005 Arabic to English and Chinese to English NIST MT evaluation tasks show significant improvements in BLEU scores compared to earlier confusion network decoding based methods.

1 Introduction

System combination has been shown to improve classification performance in various tasks. There are several approaches for combining classifiers. In ensemble learning, a collection of simple classifiers is used to yield better performance than any single classifier; for example boosting (Schapire, 1990). Another approach is to combine outputs from a few highly specialized classifiers. The classifiers may be based on the same basic modeling techniques but differ by, for example, alternative feature representations. Combination of speech recognition outputs is an example of this approach (Fiscus, 1997). In speech recognition, confusion network decoding (Mangu et al., 2000) has become widely used in system combination.

Unlike speech recognition, current statistical machine translation (MT) systems are based on various different paradigms; for example phrasal, hierarchical and syntax-based systems. The idea of combining outputs from different MT systems to produce consensus translations in the hope of generating better translations has been around for a while (Frederking and Nirenburg, 1994). Recently, confusion network decoding for MT system combination has been proposed (Bangalore et al., 2001). To generate confusion networks, hypotheses have to be aligned against each other. In (Bangalore et al., 2001), Levenshtein alignment was used to generate the network. As opposed to speech recognition, the word order between two correct MT outputs may be different and the Levenshtein alignment may not be able to align shifted words in the hypotheses. In (Matusov et al., 2006), different word orderings are taken into account by training alignment models by considering all hypothesis pairs as a parallel corpus using GIZA++ (Och and Ney, 2003). The size of the test set may influence the quality of these alignments. Thus, system outputs from development sets may have to be added to improve the GIZA++ alignments. A modified Levenshtein alignment allowing shifts as in computation of the translation edit rate (TER) (Snover et al., 2006) was used to align hy-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 312–319,
Prague, Czech Republic, June 2007.

potheses in (Sim et al., 2007). The alignments from TER are consistent as they do not depend on the test set size. Also, a more heuristic alignment method has been proposed in a different system combination approach (Jayaraman and Lavie, 2005). A full comparison of different alignment methods would be difficult as many approaches require a significant amount of engineering.

Confusion networks are generated by choosing one hypothesis as the "skeleton", and other hypotheses are aligned against it. The skeleton defines the word order of the combination output. Minimum Bayes risk (MBR) was used to choose the skeleton in (Sim et al., 2007). The average TER score was computed between each system's 1-best hypothesis and all other hypotheses. The MBR hypothesis is the one with the minimum average TER and thus, may be viewed as the closest to all other hypotheses in terms of TER. This work was extended in (Rosti et al., 2007) by introducing system weights for word confidences. However, the system weights did not influence the skeleton selection, so a hypothesis from a system with zero weight might have been chosen as the skeleton. In this work, confusion networks are generated by using the 1-best output from each system as the skeleton, and prior probabilities for each network are estimated from the average TER scores between the skeleton and other hypotheses. All resulting confusion networks are connected in parallel into a joint lattice where the prior probabilities are also multiplied by the system weights.

The combination outputs from confusion network decoding may be ungrammatical due to alignment errors. Also the word-level decoding may break coherent phrases produced by the individual systems. In this work, log-posterior probabilities are estimated for each confusion network arc instead of using votes or simple word confidences. This allows a log-linear addition of arbitrary features such as language model (LM) scores. The LM scores should increase the total log-posterior of more grammatical hypotheses. Powell's method (Brent, 1973) is used to tune the system and feature weights simultaneously so as to optimize various automatic evaluation metrics on a development set. Tuning is fully automatic, as opposed to (Matusov et al., 2006) where global system weights were set manually.

This paper is organized as follows. Three evaluation metrics used in weights tuning and reporting the test set results are reviewed in Section 2. Section 3 describes confusion network decoding for MT system combination. The extensions to add features log-linearly and improve the skeleton selection are presented in Sections 4 and 5, respectively. Section 6 details the weights optimization algorithm and the experimental results are reported in Section 7. Conclusions and future work are discussed in Section 8.

2 Evaluation Metrics

Currently, the most widely used automatic MT evaluation metric is the NIST BLEU-4 (Papineni et al., 2002). It is computed as the geometric mean of n-gram precisions up to 4-grams between the hypothesis E and reference E_r as follows

$$\mathrm{BLEU}(E, E_r) = \exp\Big(\frac{1}{4}\sum_{n=1}^{4}\log p_n(E, E_r)\Big)\gamma(E, E_r) \quad (1)$$

where $\gamma(E, E_r) \leq 1$ is the brevity penalty and $p_n(E, E_r)$ are the n-gram precisions. When multiple references are provided, the n-gram counts against all references are accumulated to compute the precisions. Similarly, full test set scores are obtained by accumulating counts over all hypothesis and reference pairs. The BLEU scores are between 0 and 1, higher being better. Often BLEU scores are reported as percentages and "one BLEU point gain" usually means a BLEU increase of 0.01.

Other evaluation metrics have been proposed to replace BLEU. It has been argued that METEOR correlates better with human judgment due to higher weight on recall than precision (Banerjee and Lavie, 2005). METEOR is based on the weighted harmonic mean of the precision and recall measured on unigram matches as follows

$$\mathrm{MTR}(E, E_r) = \frac{10m}{N_h + 9N_r}\Big(1 - 0.5(c/m)^3\Big) \quad (2)$$

where m is the total number of unigram matches, N_h is the hypothesis length, N_r is the reference length and c is the minimum number of n-gram matches that covers the alignment. The second term is a fragmentation penalty which penalizes the harmonic mean by a factor of up to 0.5 when $c = m$; i.e.,

there are no matching n-grams higher than $n = 1$. By default, METEOR script counts the words that match exactly, and words that match after a simple Porter stemmer. Additional matching modules including WordNet stemming and synonymy may also be used. When multiple references are provided, the lowest score is reported. Full test set scores are obtained by accumulating statistics over all test sentences. The METEOR scores are also between 0 and 1, higher being better. The scores in the results section are reported as percentages.

Translation edit rate (TER) (Snover et al., 2006) has been proposed as more intuitive evaluation metric since it is based on the rate of edits required to transform the hypothesis into the reference. The TER score is computed as follows

$$\mathrm{TER}(E, E_r) = \frac{\mathrm{Ins} + \mathrm{Del} + \mathrm{Sub} + \mathrm{Shft}}{N_r} \quad (3)$$

where N_r is the reference length. The only difference to word error rate is that the TER allows shifts. A shift of a sequence of words is counted as a single edit. The minimum translation edit alignment is usually found through a beam search. When multiple references are provided, the edits from the closest reference are divided by the average reference length. Full test set scores are obtained by accumulating the edits and the average reference lengths. The perfect TER score is 0, and otherwise higher than zero. The TER score may also be higher than 1 due to insertions. Also TER is reported as a percentage in the results section.

3 Confusion Network Decoding

Confusion network decoding in MT has to pick one hypothesis as the skeleton which determines the word order of the combination. The other hypotheses are aligned against the skeleton. Either votes or some form of confidences are assigned to each word in the network. For example using "cat sat the mat" as the skeleton, aligning "cat sitting on the mat" and "hat on a mat" against it might yield the following alignments:

cat	sat	ε	the	mat
cat	sitting	on	the	mat
hat	ε	on	a	mat

where ε represents a NULL word. In graphical form, the resulting confusion network is shown in Figure 1. Each arc represents an alternative word at that position in the sentence and the number of votes for each word is marked in parentheses. Confusion network decoding usually requires finding the path with the highest confidence in the network. Based on vote counts, there are three alternatives in the example: "cat sat on the mat", "cat on the mat" and "cat sitting on the mat", each having accumulated 10 votes. The alignment procedure plays an important role, as by switching the position of the word 'sat' and the following NULL in the skeleton, there would be a single highest scoring path through the network; that is, "cat on the mat".

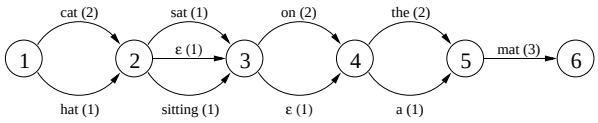


Figure 1: Example consensus network with votes on word arcs.

Different alignment methods yield different confusion networks. The modified Levenshtein alignment as used in TER is more natural than simple edit distance such as word error rate since machine translation hypotheses may have different word orders while having the same meaning. As the skeleton determines the word order, the quality of the combination output also depends on which hypothesis is chosen as the skeleton. Since the modified Levenshtein alignment produces TER scores between the skeleton and the other hypotheses, a natural choice for selecting the skeleton is the minimum average TER score. The hypothesis resulting in the lowest average TER score when aligned against all other hypotheses is chosen as the skeleton E_s as follows

$$E_s = \arg\min_{E \in E_i} \sum_{j=1}^{N_s} \mathrm{TER}(E_j, E_i) \quad (4)$$

where N_s is the number of systems. This is equivalent to minimum Bayes risk decoding with uniform posterior probabilities (Sim et al., 2007). Other evaluation metrics may also be used as the MBR loss function. For BLEU and METEOR, the loss function would be $1 - \mathrm{BLEU}(E_j, E_i)$ and $1 - \mathrm{MTR}(E_j, E_i)$.

It has been found that multiple hypotheses from each system may be used to improve the quality of

the combination output (Sim et al., 2007). When using N-best lists from each system, the words may be assigned a different score based on the rank of the hypothesis. In (Rosti et al., 2007), simple $1/(1+k)$ score was assigned to the word coming from the kth-best hypothesis. Due to the computational burden of the TER alignment, only 1-best hypotheses were considered as possible skeletons, and $n = 10$ hypotheses per system were aligned. Similar approach to estimate word posteriors is adopted in this work.

System weights may be used to assign a system specific confidence on each word in the network. The weights may be based on the systems' relative performance on a separate development set or they may be automatically tuned to optimize some evaluation metric on the development set. In (Rosti et al., 2007), the total confidence of the nth best confusion network hypothesis $E_{j,n}$, including NULL words, given the jth source sentence F_j was given by

$$c(E_{j,n}|F_j) = \sum_{i=1}^{N_j-1} \sum_{l=1}^{N_s} \lambda_l c_{wli} + \mu N_{nulls}(E_{j,n}) \quad (5)$$

where N_j is the number of nodes in the confusion network for the source sentence F_j, N_s is the number of translation systems, λ_l is the lth system weight, c_{wli} is the accumulated confidence for word w produced by system l between nodes i and $i+1$, and μ is a weight for the number of NULL links along the hypothesis $N_{nulls}(E_{j,n})$. The word confidences c_{wli} were increased by $1/(1+k)$ if the word w aligns between nodes i and $i+1$ in the network. If no word aligns between nodes i and $i+1$, the NULL word confidence at that position was increased by $1/(1+k)$. The last term controls the number of NULL words generated in the output and may be viewed as an insertion penalty. Each arc in the confusion network carries the word label w and N_s scores c_{wli}. The decoder outputs the hypothesis with the highest $c(E_{j,n}|F_j)$ given the current set of weights.

3.1 Discussion

There are several problems with the previous confusion network decoding approaches. First, the decoding can generate ungrammatical hypotheses due to alignment errors and phrases broken by the word-level decoding. For example, two synonymous words may be aligned to other words not already aligned, which may result in repetitive output. Second, the additive confidence scores in Equation 5 have no probabilistic meaning and cannot therefore be combined with language model scores. Language model expansion and re-scoring may help by increasing the probability of more grammatical hypotheses in decoding. Third, the system weights are independent of the skeleton selection. Therefore, a hypothesis from a system with a low or zero weight may be chosen as the skeleton.

4 Log-Linear Combination with Arbitrary Features

To address the issue with ungrammatical hypotheses and allow language model expansion and re-scoring, the hypothesis confidence computation is modified. Instead of summing arbitrary confidence scores as in Equation 5, word posterior probabilities are used as follows

$$\log p(E_{j,n}|F_j) = \sum_{i=1}^{N_j-1} \log \Big(\sum_{l=1}^{N_s} \lambda_l p(w|l,i) \Big) + \nu L(E_{j,n}) + \mu N_{nulls}(E_{j,n}) + \xi N_{words}(E_{j,n}) \quad (6)$$

where ν is the language model weight, $L(E_{j,n})$ is the LM log-probability and $N_{words}(E_{j,n})$ is the number of words in the hypothesis $E_{j,n}$. The word posteriors $p(w|l,i)$ are estimated by scaling the confidences c_{wli} to sum to one for each system l over all words w in between nodes i and $i+1$. The system weights are also constrained to sum to one. Equation 6 may be viewed as a log-linear sum of sentence-level features. The first feature is the sum of word log-posteriors, the second is the LM log-probability, the third is the log-NULL score and the last is the log-length score. The last two terms are not completely independent but seem to help based on experimental results.

The number of paths through a confusion network grows exponentially with the number of nodes. Therefore expanding a network with an n-gram language model may result in huge lattices if n is high. Instead of high order n-grams with heavy pruning, a bi-gram may first be used to expand the lattice. After optimizing one set of weights for the expanded

confusion network, a second set of weights for N-best list re-scoring with a higher order n-gram model may be optimized. On a test set, the first set of weights is used to generate an N-best list from the bi-gram expanded lattice. This N-best list is then re-scored with the higher order n-gram. The second set of weights is used to find the final 1-best from the re-scored N-best list.

5 Multiple Confusion Network Decoding

As discussed in Section 3, there is a disconnect between the skeleton selection and confidence estimation. To prevent the 1-best from a system with a low or zero weight being selected as the skeleton, confusion networks are generated for each system and the average TER score in Equation 4 is used to estimate a prior probability for the corresponding network. All N_s confusion networks are connected to a single start node with NULL arcs which contain the prior probability from the system used as the skeleton for that network. All confusion network are connected to a common end node with NULL arcs. The final arcs have a probability of one. The prior probabilities in the arcs leaving the first node will be multiplied by the corresponding system weights which guarantees that a path through a network generated around a 1-best from a system with a zero weight will not be chosen.

The prior probabilities are estimated by viewing the negative average TER scores between the skeleton and other hypotheses as log-probabilities. These log-probabilities are scaled so that the priors sum to one. There is a concern that the prior probabilities estimated this way may be inaccurate. Therefore, the priors may have to be smoothed by a tunable exponent. However, the optimization experiments showed that the best performance was obtained by having a smoothing factor of 1 which is equivalent to the original priors. Thus, no smoothing was used in the experiments presented later in this paper.

An example joint network with the priors is shown in Figure 2. This example has three confusion networks with priors 0.5, 0.2 and 0.3. The total number of nodes in the network is represented by N_a. Similar combination of multiple confusion networks was presented in (Matusov et al., 2006). However, this approach did not include sentence specific prior estimates, word posterior estimates, and did not allow joint optimization of the system and feature weights.

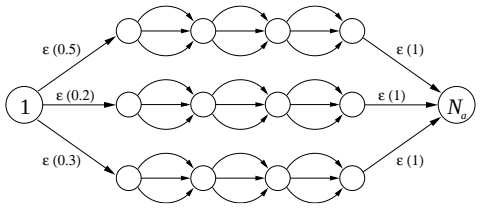


Figure 2: Three confusion networks with prior probabilities.

6 Weights Optimization

The optimization of the system and feature weights may be carried out using N-best lists as in (Ostendorf et al., 1991). A confusion network may be represented by a word lattice and standard tools may be used to generate N-best hypothesis lists including word confidence scores, language model scores and other features. The N-best list may be re-ordered using the sentence-level posteriors $p(E_{j,n}|F_j)$ from Equation 6 for the jth source sentence F_j and the corresponding nth hypothesis $E_{j,n}$. The current 1-best hypothesis $\hat{E}_j$ given a set of weights $\theta = \{\lambda_1, \ldots, \lambda_{N_s}, \nu, \mu, \xi\}$ may be represented as follows

$$\hat{E}_j(F_j|\theta) = \arg\max_{E \in E_{j,n}} p(E_{j,n}|F_j) \qquad (7)$$

The objective is to optimize the 1-best score on a development set given a set of reference translations. For example, estimating weights which minimize TER between a set of 1-best hypothesis $\hat{S}$ and reference translations S_r can be written as

$$\hat{\theta} = \arg\min_{\theta} \mathrm{TER}(\hat{S}, S_r) \qquad (8)$$

This objective function is very complicated, so gradient-based optimization methods may not be used. In this work, modified Powell's method as proposed by (Brent, 1973) is used. The algorithm explores better weights iteratively starting from a set of initial weights. First, each dimension is optimized using a grid-based line minimization algorithm. Then, a new direction based on the changes in the objective function is estimated to speed up the search. To improve the chances of finding a

global optimum, 19 random perturbations of the initial weights are used in parallel optimization runs. Since the N-best list represents only a small portion of all hypotheses in the confusion network, the optimized weights from one iteration may be used to generate a new N-best list from the lattice for the next iteration. Similarly, weights which maximize BLEU or METEOR may be optimized.

The same Powell's method has been used to estimate feature weights of a standard feature-based phrasal MT decoder in (Och, 2003). A more efficient algorithm for log-linear models was also proposed. In this work, both the system and feature weights are jointly optimized, so the efficient algorithm for the log-linear models cannot be used.

7 Results

The improved system combination method was compared to a simple confusion network decoding without system weights and the method proposed in (Rosti et al., 2007) on the Arabic to English and Chinese to English NIST MT05 tasks. Six MT systems were combined: three (A,C,E) were phrase-based similar to (Koehn, 2004), two (B,D) were hierarchical similar to (Chiang, 2005) and one (F) was syntax-based similar to (Galley et al., 2006). All systems were trained on the same data and the outputs used the same tokenization. The decoder weights for systems A and B were tuned to optimize TER, and others were tuned to optimize BLEU. All decoder weight tuning was done on the NIST MT02 task.

The joint confusion network was expanded with a bi-gram language model and a 300-best list was generated from the lattice for each tuning iteration. The system and feature weights were tuned on the union of NIST MT03 and MT04 tasks. All four reference translations available for the tuning and test sets were used. A first set of weights with the bi-gram LM was optimized with three iterations. A second set of weights was tuned for 5-gram N-best list re-scoring. The bi-gram and 5-gram English language models were trained on about 7 billion words. The final combination outputs were detokenized and cased before scoring.

The tuning set results on the Arabic to English NIST MT03+MT04 task are shown in Table 1. The best score on each metric is shown in bold face fonts. The row labeled as `no weights` corresponds to Equation 5 with uniform system weights λ_l and zero NULL weight. The `baseline` corresponds to Equation 5 with TER tuned weights. The following three rows correspond to the improved confusion network decoding with different optimization metrics. As expected, the scores on the metric used in tuning are the best on that metric. Also, the combination results are better than any single system on all metrics in the case of TER and BLEU tuning. However, the METEOR tuning yields extremely high TER and low BLEU scores. This must be due to the higher weight on the recall compared to precision in the harmonic mean used to compute the METEOR

Arabic tuning	TER	BLEU	MTR
system A	44.93	45.71	66.09
system B	46.41	43.07	64.79
system C	46.10	46.41	65.33
system D	44.36	46.83	66.91
system E	45.35	45.44	65.69
system F	47.10	44.52	65.28
no weights	42.35	48.91	67.76
baseline	42.19	49.86	68.34
TER tuned	**41.88**	51.45	68.62
BLEU tuned	42.12	**51.72**	68.59
MTR tuned	54.08	38.93	**71.42**

Table 1: Mixed-case TER and BLEU, and lower-case METEOR scores on Arabic NIST MT03+MT04.

Arabic test	TER	BLEU	MTR
system A	42.98	49.58	69.86
system B	43.79	47.06	68.62
system C	43.92	47.87	66.97
system D	40.75	52.09	71.23
system E	42.19	50.86	70.02
system F	44.30	50.15	69.75
no weights	39.33	53.66	71.61
baseline	39.29	54.51	72.20
TER tuned	**39.10**	55.30	72.53
BLEU tuned	39.13	**55.48**	72.81
MTR tuned	51.56	41.73	**74.79**

Table 2: Mixed-case TER and BLEU, and lower-case METEOR scores on Arabic NIST MT05.

Chinese tuning	TER	BLEU	MTR
system A	56.56	29.39	54.54
system B	55.88	30.45	54.36
system C	58.35	32.88	56.72
system D	57.09	36.18	57.11
system E	57.69	33.85	58.28
system F	56.11	36.64	58.90
no weights	53.11	37.77	59.19
baseline	53.40	38.52	59.56
TER tuned	**52.13**	36.87	57.30
BLEU tuned	53.03	**39.99**	58.97
MTR tuned	70.27	28.60	**63.10**

Table 3: Mixed-case TER and BLEU, and lower-case METEOR scores on Chinese NIST MT03+MT04.

Chinese test	TER	BLEU	MTR
system A	56.57	29.63	56.63
system B	56.30	29.62	55.61
system C	59.48	31.32	57.71
system D	58.32	33.77	57.92
system E	58.46	32.40	59.75
system F	56.79	35.30	60.82
no weights	53.80	36.17	60.75
baseline	54.34	36.44	61.05
TER tuned	**52.90**	35.76	58.60
BLEU tuned	54.05	**37.91**	60.31
MTR tuned	72.59	26.96	**64.35**

Table 4: Mixed-case TER and BLEU, and lower-case METEOR scores on Chinese NIST MT05.

score. Even though METEOR has been shown to be a good metric on a given MT output, tuning to optimize METEOR results in a high insertion rate and low precision. The Arabic test set results are shown in Table 2. The TER and BLEU optimized combination results beat all single system scores on all metrics. The best results on a given metric are again obtained by the combination optimized for the corresponding metric. It should be noted that the TER optimized combination has significantly higher BLEU score than the TER optimized baseline. Compared to the baseline system which is also optimized for TER, the BLEU score is improved by 0.97 points. Also, the METEOR score using the METEOR optimized weights is very high. However, the other scores are worse in common with the tuning set results.

The tuning set results on the Chinese to English NIST MT03+MT04 task are shown in Table 3. The baseline combination weights were tuned to optimize BLEU. Again, the best scores on each metric are obtained by the combination tuned for that metric. Only the METEOR score of the TER tuned combination is worse than the METEOR scores of systems E and F - other combinations are better than any single system on all metrics apart from the METEOR tuned combinations. The test set results follow clearly the tuning results again - the TER tuned combination is the best in terms of TER, the BLEU tuned in terms of BLEU, and the METEOR tuned in terms of METEOR. Compared to the baseline, the BLEU score of the BLEU tuned combination is improved by 1.47 points. Again, the METEOR tuned weights hurt the other metrics significantly.

8 Conclusions

An improved confusion network decoding method combining the word posteriors with arbitrary features was presented. This allows the addition of language model scores by expanding the lattices or re-scoring N-best lists. The LM integration should result in more grammatical combination outputs. Also, confusion networks generated by using the 1-best hypothesis from all systems as the skeleton were used with prior probabilities derived from the average TER scores. This guarantees that the best path will not be found from a network generated for a system with zero weight. Compared to the earlier system combination approaches, this method is fully automatic and requires very little additional information on top of the development set outputs from the individual systems to tune the weights.

The new method was evaluated on the Arabic to English and Chinese to English NIST MT05 tasks. Compared to the baseline from (Rosti et al., 2007), the new method improves the BLEU scores significantly. The combination weights were tuned to optimize three automatic evaluation metrics: TER, BLEU and METEOR. The TER tuning seems to yield very good results on Arabic - the BLEU tuning seems to be better on Chinese. It also seems like

METEOR should not be used in tuning due to high insertion rate and low precision. It would be interesting to know which tuning metric results in the best translations in terms of human judgment. However, this would require time consuming evaluations such as human mediated TER post-editing (Snover et al., 2006).

The improved confusion network decoding approach allows arbitrary features to be used in the combination. New features may be added in the future. Hypothesis alignment is also very important in confusion network generation. Better alignment methods which take synonymy into account should be investigated. This method could also benefit from more sophisticated word posterior estimation.

Acknowledgments

This work was supported by DARPA/IPTO Contract No. HR0011-06-C-0022 under the GALE program (approved for public release, distribution unlimited). The authors would like to thank ISI and University of Edinburgh for sharing their MT system outputs.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proc. ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72.

Srinivas Bangalore, German Bordel, and Giuseppe Riccardi. 2001. Computing consensus translation from multiple machine translation systems. In *Proc. ASRU*, pages 351–354.

Richard P. Brent. 1973. *Algorithms for Minimization Without Derivatives*. Prentice-Hall.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. ACL*, pages 263–270.

Jonathan G. Fiscus. 1997. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER). In *Proc. ASRU*, pages 347–354.

Robert Frederking and Sergei Nirenburg. 1994. Three heads are better than one. In *Proc. ANLP*, pages 95–100.

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

Shyamsundar Jayaraman and Alon Lavie. 2005. Multi-engine machine translation guided by explicit word matching. In *Proc. EAMT*, pages 143–152.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proc. AMTA*, pages 115–124.

Lidia Mangu, Eric Brill, and Andreas Stolcke. 2000. Finding consensus in speech recognition: Word error minimization and other applications of confusion networks. *Computer Speech and Language*, 14(4):373–400.

Evgeny Matusov, Nicola Ueffi ng, and Hermann Ney. 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *Proc. EACL*, pages 33–40.

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

Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. ACL*, pages 160–167.

Mari Ostendorf, Ashvin Kannan, Steve Austin, Owen Kimball, Richard Schwartz, and Jan Robin Rohlicek. 1991. Integration of diverse recognition methodologies through reevaluation of N-best sentence hypotheses. In *Proc. HLT*, pages 83–87.

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

Antti-Veikko I. Rosti, Bing Xiang, Spyros Matsoukas, Richard Schwartz, Necip Fazil Ayan, and Bonnie J. Dorr. 2007. Combining outputs from multiple machine translation systems. In *Proc. NAACL-HLT 2007*, pages 228–235.

Robert E. Schapire. 1990. The strength of weak learnability. *Machine Learning*, 5(2):197–227.

Khe Chai Sim, William J. Byrne, Mark J.F. Gales, Hichem Sahbi, and Phil C. Woodland. 2007. Consensus network decoding for statistical machine translation system combination. In *Proc. ICASSP*, volume 4, pages 105–108.

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

Generating Constituent Order in German Clauses

Katja Filippova and **Michael Strube**
EML Research gGmbH
Schloss-Wolfsbrunnenweg 33
69118 Heidelberg, Germany
http://www.eml-research.de/nlp

Abstract

We investigate the factors which determine constituent order in German clauses and propose an algorithm which performs the task in two steps: First, the best candidate for the initial sentence position is chosen. Then, the order for the remaining constituents is determined. The first task is more difficult than the second one because of properties of the German sentence-initial position. Experiments show a significant improvement over competing approaches. Our algorithm is also more efficient than these.

1 Introduction

Many natural languages allow variation in the word order. This is a challenge for natural language generation and machine translation systems, or for text summarizers. E.g., in text-to-text generation (Barzilay & McKeown, 2005; Marsi & Krahmer, 2005; Wan et al., 2005), new sentences are fused from dependency structures of input sentences. The last step of sentence fusion is linearization of the resulting parse. Even for English, which is a language with fixed word order, this is not a trivial task.

German has a relatively free word order. This concerns the order of *constituents*[1] within sentences while the order of words within constituents is relatively rigid. The grammar only partially prescribes how constituents dependent on the verb should be ordered, and for many clauses each of the $n!$ possible permutations of n constituents is grammatical.

[1]Henceforth, we will use this term to refer to constituents dependent on the clausal top node, i.e. a verb, only.

In spite of the permanent interest in German word order in the linguistics community, most studies have limited their scope to the order of verb arguments and few researchers have implemented – and even less evaluated – a generation algorithm. In this paper, we present an algorithm, which orders not only verb arguments but all kinds of constituents, and evaluate it on a corpus of biographies. For each parsed sentence in the test set, our maximum-entropy-based algorithm aims at reproducing the order found in the original text. We investigate the importance of different linguistic factors and suggest an algorithm to constituent ordering which first determines the sentence initial constituent and then orders the remaining ones. We provide evidence that the task requires language-specific knowledge to achieve better results and point to the most difficult part of it. Similar to Langkilde & Knight (1998) we utilize statistical methods. Unlike overgeneration approaches (Varges & Mellish, 2001, inter alia) which select the best of *all* possible outputs ours is more efficient, because we do not need to generate every permutation.

2 Theoretical Premises

2.1 Background

It has been suggested that several factors have an influence on German constituent order. Apart from the constraints posed by the grammar, information structure, surface form, and discourse status have also been shown to play a role. It has also been observed that there are preferences for a particular order. The preferences summarized below have mo-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 320–327,
Prague, Czech Republic, June 2007.

tivated our choice of features:

- constituents in the nominative case precede those in other cases, and dative constituents often precede those in the accusative case (Uszkoreit, 1987; Keller, 2000);
- the verb arguments' order depends on the verb's subcategorization properties (Kurz, 2000);
- constituents with a definite article precede those with an indefinite one (Weber & Müller, 2004);
- pronominalized constituents precede non-pronominalized ones (Kempen & Harbusch, 2004);
- animate referents precede inanimate ones (Pappert et al., 2007);
- short constituents precede longer ones (Kimball, 1973);
- the preferred topic position is right after the verb (Frey, 2004);
- the initial position is usually occupied by scene-setting elements and topics (Speyer, 2005).
- there is a default order based on semantic properties of constituents (Sgall et al., 1986):
  Actor < Temporal < SpaceLocative < Means < Addressee < Patient < Source < Destination < Purpose

Note that most of these preferences were identified in corpus studies and experiments with native speakers and concern the order of verb arguments only. Little has been said so far about how non-arguments should be ordered.

German is a verb second language, i.e., the position of the verb in the main clause is determined exclusively by the grammar and is insensitive to other factors. Thus, the German main clause is divided into two parts by the finite verb: *Vorfeld (VF)*, which contains exactly one constituent, and *Mittelfeld (MF)*, where the remaining constituents are located. The subordinate clause normally has only MF. The VF and MF are marked with brackets in Example 1:

(1) [Außerdem] entwickelte [Lummer eine Quecksilberdampflampe, um monochromatisches Licht herzustellen].
Apart from that developed Lummer a Mercury-vapor lamp to monochrome light produce.

'Apart from that, Lummer developed a Mercury-vapor lamp to produce monochrome light'.

2.2 Our Hypothesis

The essential contribution of our study is that we treat preverbal and postverbal parts of the sentence differently. The sentence-initial position, which in German is the VF, has been shown to be cognitively more prominent than other positions (Gernsbacher & Hargreaves, 1988). Motivated by the theoretical work by Chafe (1976) and Jacobs (2001), we view the VF as the place for elements which modify the situation described in the sentence, i.e. for so called frame-setting topics (Jacobs, 2001). For example, temporal or locational constituents, or anaphoric adverbs are good candidates for the VF. We hypothesize that the reasons which bring a constituent to the VF are different from those which place it, say, to the beginning of the MF, for the order in the MF has been shown to be relatively rigid (Keller, 2000; Kempen & Harbusch, 2004). Speakers have the freedom of selecting the outgoing point for a sentence. Once they have selected it, the remaining constituents are arranged in the MF, mainly according to their grammatical properties.

This last observation motivates another hypothesis we make: The cumulation of the properties of a constituent determines its *salience*. This salience can be calculated and used for ordering with a simple rule stating that more salient constituents should precede less salient ones. In this case there is no need to generate all possible orders and rank them. The best order can be obtained from a random one by sorting. Our experiments support this view. A two-step approach, which first selects the best candidate for the VF and then arranges the remaining constituents in the MF with respect to their salience performs better than algorithms which generate the order for a sentence as a whole.

3 Related Work

Uszkoreit (1987) addresses the problem from a mostly grammar-based perspective and suggests weighted constraints, such as [+NOM] $\prec$ [+DAT], [+PRO] $\prec$ [–PRO], [–FOCUS] $\prec$ [+FOCUS], etc.

Kruijff et al. (2001) describe an architecture which supports generating the appropriate word order for different languages. Inspired by the findings of the Prague School (Sgall et al., 1986) and Systemic Functional Linguistics (Halliday, 1985), they focus on the role that information structure plays in constituent ordering. Kruijff-Korbayová et al. (2002) address the task of word order generation in the same vein. Similar to ours, their algorithm recognizes the special role of the sentence-initial position which they reserve for the *theme* – the point of departure of the message. Unfortunately, they did not implement their algorithm, and it is hard to judge how well the system would perform on real data.

Harbusch et al. (2006) present a generation workbench, which has the goal of producing not the most appropriate order, but all grammatical ones. They also do not provide experimental results.

The work of Uchimoto et al. (2000) is done on the free word order language Japanese. They determine the order of phrasal units dependent on the same modifiee. Their approach is similar to ours in that they aim at regenerating the original order from a dependency parse, but differs in the scope of the problem as they regenerate the order of modifers for all and not only for the top clausal node. Using a maximum entropy framework, they choose the most probable order from the set of all permutations of n words by the following formula:

$$\begin{aligned} P(1|h) &= P(\{W_{i,i+j} = 1 | 1 \leq i \leq n-1, 1 \leq j \leq n-i\}|h) \\ &\approx \prod_{i=1}^{n-1} \prod_{j=1}^{n-i} P(W_{i,i+j} = 1 | h_{i,i+j}) \\ &= \prod_{i=1}^{n-1} \prod_{j=1}^{n-i} P_{ME}(1 | h_{i,i+j}) \end{aligned} \quad (1)$$

For each permutation, for every pair of words , they multiply the probability of their being in the correct[2] order given the history h. Random variable $W_{i,i+j}$ is 1 if word w_i precedes w_{i+j} in the reference sentence, 0 otherwise. The features they use are akin to those which play a role in determining German word order. We use their approach as a non-trivial baseline in our study.

Ringger et al. (2004) aim at regenerating the order of constituents as well as the order within them for German and French technical manuals. Utilizing syntactic, semantic, sub-categorization and length features, they test several statistical models to find the order which maximizes the probability of an ordered tree. Using "Markov grammars" as the starting point and conditioning on the syntactic category only, they expand a non-terminal node C by predicting its daughters from left to right:

$$P(C|h) = \prod_{i=1}^{n} P(d_i | d_{i-1}, ..., d_{i-j}, c, h) \quad (2)$$

Here, c is the syntactic category of C, d and h are the syntactic categories of C's daughters and the daughter which is the head of C respectively.

In their simplest system, whose performance is only 2.5% worse than the performance of the best one, they condition on both syntactic categories and semantic relations (ψ) according to the formula:

$$P(C|h) = \prod_{i=1}^{n} \left[\begin{array}{l} P(\psi_i | d_{i-1}, \psi_{i-1}, ...d_{i-j}, \psi_{i-j}, c, h) \\ \times P(d_i | \psi_i, d_{i-1}, \psi_{i-1}..., d_{i-j}, \psi_{i-j}, c, h) \end{array} \right] \quad (3)$$

Although they test their system on German data, it is hard to compare their results to ours directly. First, the metric they use does not describe the performance appropriately (see Section 6.1). Second, while the word order within NPs and PPs as well as the verb position are prescribed by the grammar to a large extent, the constituents can theoretically be ordered in any way. Thus, by generating the order for every non-terminal node, they combine two tasks of different complexity and mix the results of the more difficult task with those of the easier one.

4 Data

The data we work with is a collection of biographies from the German version of Wikipedia[3]. Fully automatic preprocessing in our system comprises the following steps: First, a list of people of a certain Wikipedia category is taken and an article is extracted for every person. Second, sentence

[2] Only reference orders are assumed to be correct.

[3] http://de.wikipedia.org

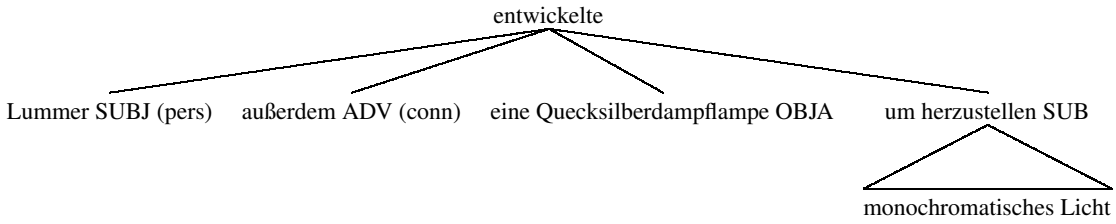


Figure 1: The representation of the sentence in Example 1

boundaries are identified with a Perl CPAN module[4] whose performance we improved by extending the list of abbreviations. Next, the sentences are split into tokens. The TnT tagger (Brants, 2000) and the TreeTagger (Schmid, 1997) are used for tagging and lemmatization. Finally, the articles are parsed with the CDG dependency parser (Foth & Menzel, 2006). Named entities are classified according to their semantic type using lists and category information from Wikipedia: person (*pers*), location (*loc*), organization (*org*), or undefined named entity (*undef_ne*). Temporal expressions (*Oktober 1915*, *danach (after that)* etc.) are identified automatically by a set of patterns. Inevitable during automatic annotation, errors at one of the preprocessing stages cause errors at the ordering stage.

Distinguishing between main and subordinate clauses, we split the total of about 19 000 sentences into training, development and test sets (Table 1). Clauses with one constituent are sorted out as trivial. The distribution of both types of clauses according to their length in constituents is given in Table 2.

	train	dev	test
main	14324	3344	1683
sub	3304	777	408
total	17628	4121	2091

Table 1: Size of the data sets in clauses

	2	3	4	5	6+
main	20%	35%	27%	12%	6%
sub	49%	35%	11%	2%	3%

Table 2: Proportion of clauses with certain lengths

[4]http://search.cpan.org/~holsten/Lingua-DE-Sentence-0.07/Sentence.pm

Given the sentence in Example 1, we first transform its dependency parse into a more general representation (Figure 1[5]) and then, based on the predictions of our learner, arrange the four constituents. For evaluation, we compare the arranged order against the original one.

Note that we predict neither the position of the verb, nor the order within constituents as the former is explicitly determined by the grammar, and the latter is much more rigid than the order of constituents.

5 Baselines and Algorithms

We compare the performance of two our algorithms with four baselines.

5.1 Random

We improve a trivial random baseline (RAND) by two syntax-oriented rules: the first position is reserved for the subject and the second for the direct object if there is any; the order of the remaining constituents is generated randomly (RAND_IMP).

5.2 Statistical Bigram Model

Similar to Ringger et al. (2004), we find the order with the highest probability conditioned on syntactic and semantic categories. Unlike them we use dependency parses and compute the probability of the top node only, which is modified by all constituents. With these adjustments the probability of an order O given the history h, if conditioned on syntactic functions of constituents $(s_1 ... s_n)$, is simply:

$$P(O|h) = \prod_{i=1}^{n} P(s_i|s_{i-1}, h) \qquad (4)$$

Ringger et al. (2004) do not make explicit, what their set of semantic relations consists of. From the

[5]OBJA stands for the accusative object.

example in the paper, it seems that these are a mixture of lexical and syntactic information[6]. Our annotation does not specify semantic relations. Instead, some of the constituents are categorized as *pers, loc, temp, org* or *undef_ne* if their heads bear one of these labels. By joining these with possible syntactic functions, we obtain a larger set of syntactic-semantic tags as, e.g., *subj-pers, pp-loc, adv-temp*. We transform each clause in the training set into a sequence of such tags, plus three tags for the verb position (*v*), the beginning (*b*) and the end (*e*) of the clause. Then we compute the bigram probabilities[7].

For our third baseline (BIGRAM), we select from all possible orders the one with the highest probability as calculated by the following formula:

$$P(O|h) = \prod_{i=1}^{n} P(t_i|t_{i-1}, h) \quad (5)$$

where t_i is from the set of joined tags. For Example 1, possible tag sequences (i.e. orders) are *'b subj-pers v adv obja sub e', 'b adv v subj-pers obja sub e', 'b obja v adv sub subj-pers e'*, etc.

5.3 Uchimoto

For the fourth baseline (UCHIMOTO), we utilized a maximum entropy learner (OpenNLP[8]) and reimplemented the algorithm of Uchimoto et al. (2000). For every possible permutation, its probability is estimated according to Formula (1). The binary classifier, whose task was to predict the probability that the order of a pair of constituents is correct, was trained on the following features describing the verb or h_c – the head of a constituent c[9]:

vlex, vpass, vmod the lemma of the root of the clause (non-auxiliary verb), the voice of the verb and the number of constituents to order;

lex the lemma of h_c or, if h_c is a functional word, the lemma of the word which depends on it;

pos part-of-speech tag of h_c;

sem if defined, the semantic class of *c*; e.g. *im April 1900* and *mit Albert Einstein (with Albert Einstein)* are classified *temp* and *pers* respectively;

syn, same the syntactic function of h_c and whether it is the same for the two constituents;

mod number of modifiers of h_c;

rep whether h_c appears in the preceding sentence;

pro whether *c* contains a (anaphoric) pronoun.

5.4 Maximum Entropy

The first configuration of our system is an extended version of the UCHIMOTO baseline (MAXENT). To the features describing c we added the following ones:

det the kind of determiner modifying h_c (*def, indef, non-appl*);

rel whether h_c is modified by a relative clause (*yes, no, non-appl*);

dep the depth of *c*;

len the length of *c* in words.

The first two features describe the discourse status of a constituent; the other two provide information on its "weight". Since our learner treats all values as nominal, we discretized the values of **dep** and **len** with a C4.5 classifier (Kohavi & Sahami, 1996).

Another modification concerns the efficiency of the algorithm. Instead of calculating probabilities for all pairs, we obtain the right order from a random one by *sorting*. We compare adjacent elements by consulting the learner as if we would sort an array of numbers. Given two adjacent constituents, $c_i < c_j$, we check the probability of their being in the right order, i.e. that c_i precedes c_j: $P_{pre}(c_i, c_j)$. If it is less than 0.5, we transpose the two and compare c_i with the next one.

Since the sorting method presupposes that the predicted relation is transitive, we checked whether this is really so on the development and test data sets. We looked for three constituents c_i, c_j, c_k from a sentence S, such that $P_{pre}(c_i, c_j) > 0.5, P_{pre}(c_j, c_k) > 0.5, P_{pre}(c_i, c_k) < 0.5$ and found none. Therefore, unlike UCHIMOTO, where one needs to make exactly $N! * N(N-1)/2$ comparisons, we have to make $N(N-1)/2$ comparisons at most.

[6] E.g. *DefDet, Coords, Possr, werden*

[7] We use the CMU Toolkit (Clarkson & Rosenfeld, 1997).

[8] http://opennlp.sourceforge.net

[9] We disregarded features which use information specific to Japanese and non-applicable to German (e.g. on postpositional particles).

5.5 The Two-Step Approach

The main difference between our first algorithm (MAXENT) and the second one (TWO-STEP) is that we generate the order in two steps[10] (both classifiers are trained on the same features):

1. For the VF, using the OpenNLP maximum entropy learner for a binary classification (VF vs. MF), we select the constituent c with the highest probability of being in the VF.

2. For the MF, the remaining constituents are put into a random order and then *sorted* the way it is done for MAXENT. The training data for the second task was generated only from the MF of clauses.

6 Results

6.1 Evaluation Metrics

We use several metrics to evaluate our systems and the baselines. The first is per-sentence *accuracy* (*acc*) which is the proportion of correctly regenerated sentences. Kendall's τ, which has been used for evaluating sentence ordering tasks (Lapata, 2006), is the second metric we use. τ is calculated as $1 - 4\frac{t}{N(N-1)}$, where t is the number of interchanges of consecutive elements to arrange N elements in the right order. τ is sensitive to near misses and assigns *abdc* (almost correct order) a score of 0.66 while *dcba* (inverse order) gets -1. Note that it is questionable whether this metric is as appropriate for word ordering tasks as for sentence ordering ones because a near miss might turn out to be ungrammatical whereas a more different order stays acceptable.

Apart from *acc* and τ, we also adopt the metrics used by Uchimoto et al. (2000) and Ringger et al. (2004). The former use *agreement rate* (*agr*) calculated as $\frac{2p}{N(N-1)}$: the number of correctly ordered pairs of constituents over the total number of all possible pairs, as well as *complete agreement* which is basically per-sentence accuracy. Unlike τ, which has -1 as the lowest score, *agr* ranges from 0 to 1. Ringger et al. (2004) evaluate the performance only in terms of *per-constituent edit distance* calculated as $\frac{m}{N}$, where m is the minimum number of moves[11] needed to arrange N constituents in the right order. This measure seems less appropriate than τ or *agr* because it does not take the distance of the move into account and scores *abced* and *eabcd* equally (0.2).

Since τ and *agr*, unlike *edit distance*, give higher scores to better orders, we compute *inverse distance*: *inv = 1 – edit_distance* instead. Thus, all three metrics (τ, *agr, inv*) give the maximum of 1 if constituents are ordered correctly. However, like τ, *agr* and *inv* can give a positive score to an ungrammatical order. Hence, none of the evaluation metrics describes the performance perfectly. Human evaluation which reliably distinguishes between appropriate, acceptable, grammatical and ingrammatical orders was out of choice because of its high cost.

6.2 Results

The results on the test data are presented in Table 3. The performance of TWO-STEP is significantly better than any other method (χ^2, $p < 0.01$). The performance of MAXENT does not significantly differ from UCHIMOTO. BIGRAM performed about as good as UCHIMOTO and MAXENT. We also checked how well TWO-STEP performs on each of the two sub-tasks (Table 4) and found that the VF selection is considerably more difficult than the sorting part.

	acc	τ	agr	inv
RAND	15%	0.02	0.51	0.64
RAND_IMP	23%	0.24	0.62	0.71
BIGRAM	51%	0.60	0.80	0.83
UCHIMOTO	50%	0.65	0.82	0.83
MAXENT	52%	0.67	0.84	0.84
TWO-STEP	61%	0.72	0.86	0.87

Table 3: Per-clause mean of the results

The most important conclusion we draw from the results is that the gain of 9% accuracy is due to the VF selection only, because the feature sets are identical for MAXENT and TWO-STEP. From this follows that doing feature selection without splitting the task in two is ineffective, because the importance of a feature depends on whether the VF or the MF is considered. For the MF, feature selection has shown **syn** and **pos** to be the most relevant features. They alone bring the performance in the MF up to 75%. In contrast, these two features explain only 56% of the

[10] Since subordinate clauses do not have a VF, the first step is not needed.

[11] A move is a deletion combined with an insertion.

cases in the VF. This implies that the order in the MF mainly depends on grammatical features, while for the VF all features are important because removal of any feature caused a loss in accuracy.

	acc	τ	agr	inv
TWO-STEP VF	68%	-	-	-
TWO-STEP MF	80%	0.92	0.96	0.95

Table 4: Mean of the results for the VF and the MF

Another important finding is that there is no need to overgenerate to find the right order. Insignificant for clauses with two or three constituents, for clauses with 10 constituents, the number of comparisons is reduced drastically from 163,296,000 to 45.

According to the *inv* metric, our results are considerably worse than those reported by Ringger et al. (2004). As mentioned in Section 3, the fact that they generate the order for every non-terminal node seriously inflates their numbers. Apart from that, they do not report accuracy, and it is unknown, how many sentences they actually reproduced correctly.

6.3 Error Analysis

To reveal the main error sources, we analyzed incorrect predictions concerning the VF and the MF, one hundred for each. Most errors in the VF did not lead to unacceptability or ungrammaticality. From lexical and semantic features, the classifier learned that some expressions are often used in the beginning of a sentence. These are temporal or locational PPs, anaphoric adverbials, some connectives or phrases starting with *unlike X, together with X, as X*, etc. Such elements were placed in the VF instead of the subject and caused an error although both variants were equally acceptable. In other cases the classifier could not find a better candidate but the subject because it could not conclude from the provided features that another constituent would nicely introduce the sentence into the discourse. Mainly this concerns recognizing information familiar to the reader not by an already mentioned entity, but one which is inferrable from what has been read.

In the MF, many orders had a PP transposed with the direct object. In some cases the predicted order seemed as good as the correct one. Often the algorithm failed at identifying verb-specific preferences: E.g., some verbs take PPs with the locational meaning as an argument and normally have them right next to them, whereas others do not. Another frequent error was the wrong placement of superficially identical constituents, e.g. two PPs of the same size. To handle this error, the system needs more specific semantic information. Some errors were caused by the parser, which created extra constituents (e.g. false PP or adverb attachment) or confused the subject with the direct verb.

We retrained our system on a corpus of newspaper articles (Telljohann et al., 2003, TüBa-D/Z) which is manually annotated but encodes no semantic knowledge. The results for the MF were the same as on the data from Wikipedia. The results for the VF were much worse (45%) because of the lack of semantic information.

7 Conclusion

We presented a novel approach to ordering constituents in German. The results indicate that a linguistically-motivated two-step system, which first selects a constituent for the initial position and then orders the remaining ones, works significantly better than approaches which do not make this separation. Our results also confirm the hypothesis – which has been attested in several corpus studies – that the order in the MF is rather rigid and dependent on grammatical properties.

We have also demonstrated that there is no need to overgenerate to find the best order. On a practical side, this finding reduces the amount of work considerably. Theoretically, it lets us conclude that the relatively fixed order in the MF depends on the salience which can be predicted mainly from grammatical features. It is much harder to predict which element should be placed in the VF. We suppose that this difficulty comes from the double function of the initial position which can either introduce the addressation topic, or be the scene- or frame-setting position (Jacobs, 2001).

Acknowledgements: This work has been funded by the Klaus Tschira Foundation, Heidelberg, Germany. The first author has been supported by a KTF grant (09.009.2004). We would also like to thank Elke Teich and the three anonymous reviewers for their useful comments.

References

Barzilay, R. & K. R. McKeown (2005). Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–327.

Brants, T. (2000). TnT – A statistical Part-of-Speech tagger. In *Proceedings of the 6th Conference on Applied Natural Language Processing,* Seattle, Wash., 29 April – 4 May 2000, pp. 224–231.

Chafe, W. (1976). Givenness, contrastiveness, definiteness, subjects, topics, and point of view. In C. Li (Ed.), *Subject and Topic*, pp. 25–55. New York, N.Y.: Academic Press.

Clarkson, P. & R. Rosenfeld (1997). Statistical language modeling using the CMU-Cambridge toolkit. In *Proceedings of the 5th European Conference on Speech Communication and Technology,* Rhodes, Greece, 22-25 September 1997, pp. 2707–2710.

Foth, K. & W. Menzel (2006). Hybrid parsing: Using probabilistic models as predictors for a symbolic parser. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics,* Sydney, Australia, 17–21 July 2006, pp. 321–327.

Frey, W. (2004). A medial topic position for German. *Linguistische Berichte*, 198:153–190.

Gernsbacher, M. A. & D. J. Hargreaves (1988). Accessing sentence participants: The advantage of first mention. *Journal of Memory and Language*, 27:699–717.

Halliday, M. A. K. (1985). *Introduction to Functional Grammar*. London, UK: Arnold.

Harbusch, K., G. Kempen, C. van Breugel & U. Koch (2006). A generation-oriented workbench for performance grammar: Capturing linear order variability in German and Dutch. In *Proceedings of the International Workshop on Natural Language Generation,* Sydney, Australia, 15-16 July 2006, pp. 9–11.

Jacobs, J. (2001). The dimensions of topic-comment. *Linguistics*, 39(4):641–681.

Keller, F. (2000). *Gradience in Grammar: Experimental and Computational Aspects of Degrees of Grammaticality*, (Ph.D. thesis). University of Edinburgh.

Kempen, G. & K. Harbusch (2004). How flexible is constituent order in the midfield of German subordinate clauses? A corpus study revealing unexpected rigidity. In *Proceedings of the International Conference on Linguistic Evidence,* Tübingen, Germany, 29–31 January 2004, pp. 81–85.

Kimball, J. (1973). Seven principles of surface structure parsing in natural language. *Cognition*, 2:15–47.

Kohavi, R. & M. Sahami (1996). Error-based and entropy-based discretization of continuous features. In *Proceedings of the 2nd International Conference on Data Mining and Knowledge Discovery,* Portland, Oreg., 2–4 August, 1996, pp. 114–119.

Kruijff, G.-J., I. Kruijff-Korbayová, J. Bateman & E. Teich (2001). Linear order as higher-level decision: Information structure in strategic and tactical generation. In *Proceedings of the 8th European Workshop on Natural Language Generation,* Toulouse, France, 6-7 July 2001, pp. 74–83.

Kruijff-Korbayová, I., G.-J. Kruijff & J. Bateman (2002). Generation of appropriate word order. In K. van Deemter & R. Kibble (Eds.), *Information Sharing: Reference and Presupposition in Language Generation and Interpretation*, pp. 193–222. Stanford, Cal.: CSLI.

Kurz, D. (2000). A statistical account on word order variation in German. In A. Abeillé, T. Brants & H. Uszkoreit (Eds.), *Proceedings of the COLING Workshop on Linguistically Interpreted Corpora,* Luxembourg, 6 August 2000.

Langkilde, I. & K. Knight (1998). Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics,* Montréal, Québec, Canada, 10–14 August 1998, pp. 704–710.

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

Marsi, E. & E. Krahmer (2005). Explorations in sentence fusion. In *Proceedings of the European Workshop on Natural Language Generation,* Aberdeen, Scotland, 8–10 August, 2005, pp. 109–117.

Pappert, S., J. Schliesser, D. P. Janssen & T. Pechmann (2007). Corpus- and psycholinguistic investigations of linguistic constraints on German word order. In A. Steube (Ed.), *The discourse potential of underspecified structures: Event structures and information structures*. Berlin, New York: Mouton de Gruyter. In press.

Ringger, E., M. Gamon, R. C. Moore, D. Rojas, M. Smets & S. Corston-Oliver (2004). Linguistically informed statistical models of constituent structure for ordering in sentence realization. In *Proceedings of the 20th International Conference on Computational Linguistics,* Geneva, Switzerland, 23–27 August 2004, pp. 673–679.

Schmid, H. (1997). Probabilistic Part-of-Speech tagging using decision trees. In D. Jones & H. Somers (Eds.), *New Methods in Language Processing*, pp. 154–164. London, UK: UCL Press.

Sgall, P., E. Hajičová & J. Panevová (1986). *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. Dordrecht, The Netherlands: D. Reidel.

Speyer, A. (2005). Competing constraints on Vorfeldbesetzung in German. In *Proceedings of the Constraints in Discourse Workshop,* Dortmund, 3–5 July 2005, pp. 79–87.

Telljohann, H., E. W. Hinrichs & S. Kübler (2003). *Stylebook for the Tübingen treebank of written German (TüBa-D/Z.* Technical Report: Seminar für Sprachwissenschaft, Universität Tübingen, Tübingen, Germany.

Uchimoto, K., M. Murata, Q. Ma, S. Sekine & H. Isahara (2000). Word order acquisition from corpora. In *Proceedings of the 18th International Conference on Computational Linguistics,* Saarbrücken, Germany, 31 July – 4 August 2000, pp. 871–877.

Uszkoreit, H. (1987). *Word Order and Constituent Structure in German*. CSLI Lecture Notes. Stanford: CSLI.

Varges, S. & C. Mellish (2001). Instance-based natural language generation. In *Proceedings of the 2nd Conference of the North American Chapter of the Association for Computational Linguistics,* Pittsburgh, Penn., 2–7 June, 2001, pp. 1–8.

Wan, S., R. Dale, M. Dras & C. Paris (2005). Searching for grammaticality and consistency: Propagating dependencies in the Viterbi algorithm. In *Proceedings of the 10th European Workshop on Natural Language Generation,* Aberdeen, Scotland*, 8–10 August, 2005*, pp. 211–216.

Weber, A. & K. Müller (2004). Word order variation in German main clauses: A corpus analysis. In *Proceedings of the 5th International Workshop on Linguistically Interpreted Corpora,* 29 August, 2004, Geneva, Switzerland, pp. 71–77.

A Symbolic Approach to Near-Deterministic Surface Realisation using Tree Adjoining Grammar

Claire Gardent
CNRS/LORIA
Nancy, France
claire.gardent@loria.fr

Eric Kow
INRIA/LORIA/UHP
Nancy, France
eric.kow@loria.fr

Abstract

Surface realisers divide into those used in generation (NLG geared realisers) and those mirroring the parsing process (Reversible realisers). While the first rely on grammars not easily usable for parsing, it is unclear how the second type of realisers could be parameterised to yield from among the set of possible paraphrases, the paraphrase appropriate to a given generation context. In this paper, we present a surface realiser which combines a reversible grammar (used for parsing and doing semantic construction) with a symbolic means of selecting paraphrases.

1 Introduction

In generation, the *surface realisation* task consists in mapping a semantic representation into a grammatical sentence.

Depending on their use, on their degree of non-determinism and on the type of grammar they assume, existing surface realisers can be divided into two main categories namely, *NLG (Natural Language Generation) geared realisers* and *reversible realisers*.

NLG geared realisers are meant as modules in a full-blown generation system and as such, they are constrained to be deterministic: a generation system must output exactly one text, no less, no more. In order to ensure this determinism, NLG geared realisers generally rely on theories of grammar which systematically link form to function such as systemic functional grammar (SFG, (Matthiessen and Bateman, 1991)) and, to a lesser extent, Meaning Text Theory (MTT, (Mel'cuk, 1988)). In these theories, a sentence is associated not just with a semantic representation but with a semantic representation enriched with additional syntactic, pragmatic and/or discourse information. This additional information is then used to constrain the realiser output.[1] One drawback of these NLG geared realisers however, is that the grammar used is not usually reversible i.e., cannot be used both for parsing and for generation. Given the time and expertise involved in developing a grammar, this is a non-trivial drawback.

Reversible realisers on the other hand, are meant to mirror the parsing process. They are used on a grammar developed for parsing and equipped with a compositional semantics. Given a string and such a grammar, a parser will assign the input string all the semantic representations associated with that string by the grammar. Conversely, given a semantic representation and the same grammar, a realiser will assign the input semantics all the strings associated with that semantics by the grammar. In such approaches, non-determinism is usually handled by statistical filtering: treebank induced probabilities are used to select from among the possible paraphrases, the most probable one. Since the most probable paraphrase is not necessarily the most appropriate one in a given context, it is unclear however, how such realisers could be integrated into a generation system.

In this paper, we present a surface realiser which

[1] On the other hand, one of our reviewers noted that "determinism" often comes more from defaults when input constraints are not supplied. One might see these realisers as being less deterministic than advertised; however, the point is that it is possible to supply the constraints that ensure determinism.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 328–335,
Prague, Czech Republic, June 2007.

combines reversibility with a symbolic approach to determinism. The grammar used is fully reversible (it is used for parsing) and the realisation algorithm can be constrained by the input so as to ensure a unique output conforming to the requirement of a given (generation) context. We show both that the grammar used has a good paraphrastic power (it is designed in such a way that grammatical paraphrases are assigned the same semantic representations) and that the realisation algorithm can be used either to generate all the grammatical paraphrases of a given input or just one provided the input is adequately constrained.

The paper is structured as follows. Section 2 introduces the grammar used namely, a Feature Based Lexicalised Tree Adjoining Grammar enriched with a compositional semantics. Importantly, this grammar is compiled from a more abstract specification (a so-called "meta-grammar") and as we shall see, it is this feature which permits a natural and systematic coupling of semantic literals with syntactic annotations. Section 3 defines the surface realisation algorithm used to generate sentences from semantic formulae. This algorithm is non-deterministic and produces all paraphrases associated by the grammar with the input semantics. We then go on to show (section 4) how this algorithm can be used on a semantic input enriched with syntactic or more abstract control annotations and further, how these annotations can be used to select from among the set of admissible paraphrases precisely these which obey the constraints expressed in the added annotations. Section 5 reports on a quantitative evaluation based on the use of a core tree adjoining grammar for French. The evaluation gives an indication of the paraphrasing power of the grammar used as well as some evidence of the deterministic nature of the realiser. Section 6 relates the proposed approach to existing work and section 7 concludes with pointers for further research.

2 The grammar

We use a unification based version of LTAG namely, Feature-based TAG. A Feature-based TAG (FTAG, (Vijay-Shanker and Joshi, 1988)) consists of a set of (auxiliary or initial) elementary trees and of two tree composition operations: substitution and adjunction. Initial trees are trees whose leaves are labelled with substitution nodes (marked with a downarrow) or terminal categories. Auxiliary trees are distinguished by a foot node (marked with a star) whose category must be the same as that of the root node. Substitution inserts a tree onto a substitution node of some other tree while adjunction inserts an auxiliary tree into a tree. In an FTAG, the tree nodes are furthermore decorated with two feature structures (called **top** and **bottom**) which are unified during derivation as follows. On substitution, the top of the substitution node is unified with the top of the root node of the tree being substituted in. On adjunction, the top of the root of the auxiliary tree is unified with the top of the node where adjunction takes place; and the bottom features of the foot node are unified with the bottom features of this node. At the end of a derivation, the top and bottom of all nodes in the derived tree are unified.

To associate semantic representations with natural language expressions, the FTAG is modified as proposed in (Gardent and Kallmeyer, 2003).

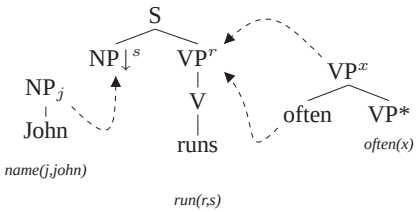


Figure 1: Flat Semantics for "John often runs"

Each elementary tree is associated with a flat semantic representation. For instance, in Figure 1,[2] the trees for *John*, *runs* and *often* are associated with the semantics *name(j,john)*, *run(r,s)* and *often(x)* respectively.

Importantly, the arguments of a semantic functor are represented by unification variables which occur both in the semantic representation of this functor and on some nodes of the associated syntactic tree. For instance in Figure 1, the semantic index s occurring in the semantic representation of *runs* also occurs on the subject substitution node of the associated elementary tree.

[2] C^x/C_x abbreviate a node with category C and a top/bottom feature structure including the feature-value pair { **index :** x}.

The value of semantic arguments is determined by the unifications resulting from adjunction and substitution. For instance, the semantic index s in the tree for *runs* is unified during substitution with the semantic indices labelling the root nodes of the tree for *John*. As a result, the semantics of *John often runs* is

(1) `{name(j,john),run(r,j),often(r)}`

The grammar used describes a core fragment of French and contains around 6 000 elementary trees. It covers some 35 basic subcategorisation frames and for each of these frames, the set of argument redistributions (active, passive, middle, neuter, reflexivisation, impersonal, passive impersonal) and of argument realisations (cliticisation, extraction, omission, permutations, etc.) possible for this frame. As a result, it captures most grammatical paraphrases that is, paraphrases due to diverging argument realisations or to different meaning preserving alternation (e.g., active/passive or clefted/non-clefted sentence).

3 The surface realiser, GenI

The basic surface realisation algorithm used is a bottom up, tabular realisation algorithm (Kay, 1996) optimised for TAGs. It follows a three step strategy which can be summarised as follows. Given an empty agenda, an empty chart and an input semantics ϕ:

Lexical selection. Select all elementary trees whose semantics subsumes (part of) ϕ. Store these trees in the agenda. Auxiliary trees devoid of substitution nodes are stored in a separate agenda called the auxiliary agenda.

Substitution phase. Retrieve a tree from the agenda, add it to the chart and try to combine it by substitution with trees present in the chart. Add any resulting derived tree to the agenda. Stop when the agenda is empty.

Adjunction phase. Move the chart trees to the agenda and the auxiliary agenda trees to the chart. Retrieve a tree from the agenda, add it to the chart and try to combine it by adjunction with trees present in the chart. Add any resulting derived tree to the agenda. Stop when the agenda is empty.

When processing stops, the yield of any syntactically complete tree whose semantics is ϕ yields an output i.e., a sentence.

The workings of this algorithm can be illustrated by the following example. Suppose that the input semantics is (1). In a first step (**lexical selection**), the elementary trees selected are the ones for *John*, *runs*, *often*. Their semantics subsumes part of the input semantics. The trees for *John* and *runs* are placed on the agenda, the one for *often* is placed on the auxiliary agenda.

The second step (the **substitution phase**) consists in systematically exploring the possibility of combining two trees by substitution. Here, the tree for *John* is substituted into the one for *runs*, and the resulting derived tree for *John runs* is placed on the agenda. Trees on the agenda are processed one by one in this fashion. When the agenda is empty, indicating that all combinations have been tried, we prepare for the next phase.

All items containing an empty substitution node are erased from the chart (here, the tree anchored by *runs*). The agenda is then reinitialised to the content of the chart and the chart to the content of the auxiliary agenda (here *often*). The **adjunction phase** proceeds much like the previous phase, except that now all possible adjunctions are performed. When the agenda is empty once more, the items in the chart whose semantics matches the input semantics are selected, and their strings printed out, yielding in this case the sentence *John often runs*.

4 Paraphrase selection

The surface realisation algorithm just sketched is non-deterministic. Given a semantic formula, it might produce several outputs. For instance, given the appropriate grammar for French, the input in (2a) will generate the set of paraphrases partly given in (2b-2k).

(2) a. l_j*:jean(j)* l_a*:aime(e,j,m)* l_m*:marie(m)*
b. Jean aime Marie
c. Marie est aimée par Jean
d. C'est Jean qui aime Marie
e. C'est Jean par qui Marie est aimée
f. C'est par Jean qu'est aimée Marie
g. C'est Jean dont est aimée Marie
h. C'est Jean dont Marie est aimée
i. C'est Marie qui est aimée par Jean

j. C'est Marie qu'aime Jean

k. C'est Marie que Jean aime

To select from among all possible paraphrases of a given input, exactly one paraphrase, NLG geared realisers use symbolic information to encode syntactic, stylistic or pragmatic constraints on the output. Thus for instance, both REALPRO (Lavoie and Rambow, 1997) and SURGE (Elhadad and Robin, 1999) assume that the input associates semantic literals with low level syntactic and lexical information mostly leaving the realiser to just handle inflection, word order, insertion of grammatical words and agreement. Similarly, KPML (Matthiessen and Bateman, 1991) assumes access to ideational, interpersonal and textual information which roughly corresponds to semantic, mood/voice, theme/rheme and focus/ground information.

In what follows, we first show that the semantic input assumed by the realiser sketched in the previous section can be systematically enriched with syntactic information so as to ensure determinism. We then indicate how the satisfiability of this enriched input could be controlled.

4.1 At most one realisation

In the realisation algorithm sketched in Section 3, non-determinism stems from lexical ambiguity:[3] for each (combination of) literal(s) l in the input there usually is more than one TAG elementary tree whose semantics subsumes l. Thus each (combination of) literal(s) in the input selects a set of elementary trees and the realiser output is the set of combinations of selected lexical trees which are licensed by the grammar operations (substitution and adjunction) and whose semantics is the input.

One way to enforce determinism consists in ensuring that each literal in the input selects exactly one elementary tree. For instance, suppose we want to generate (2b), repeated here as (3a), rather than any of the paraphrases listed in (2c-2k). Intuitively, the syntactic constraints to be expressed are those given in (3b).

(3) a. Jean aime Marie

b. Canonical Nominal Subject, Active verb form, Canonical Nominal Object

c. l_j*:jean(j)* l_a*:aime(e,j,m)* l_m*:marie(m)*

The question is how precisely to formulate these constraints, how to associate them with the semantic input assumed in Section 3 and how to ensure that the constraints used do enforce uniqueness of selection (i.e., that for each input literal, exactly one elementary tree is selected)? To answer this, we rely on a feature of the grammar used, namely that *each elementary tree is associated with a linguistically meaningful unique identifier.*

The reason for this is that the grammar is compiled from a higher level description where tree fragments are first encapsulated into so-called classes and then explicitly combined (by inheritance, conjunction and disjunction) to produce the grammar elementary trees (cf. (Crabbé and Duchier, 2004)).

More generally, each elementary tree in the grammar is associated with the set of classes used to produce that tree and importantly, this set of classes (we will call this the *tree identifier*) provides a distinguishing description (a unique identifier) for that tree: a tree is defined by a specific combination of classes and conversely, a specific combination of classes yields a unique tree.[4] Thus the set of classes associated by the compilation process with a given elementary tree can be used to uniquely identify that tree.

Given this, surface realisation is constrained as follows.

1. Each tree identifier *Id(tree)* is mapped into a simplified set of tree properties TP_t. There are two reasons for this simplification. First, some classes are irrelevant. For instance, the class used to enforce subject-verb agreement is needed to ensure this agreement but does not help in selecting among competing trees. Second, a given class C can be defined to be

[3] Given two TAG trees, there might also be several ways of combining them thereby inducing more non-determinism. However in practice we found that most of this non-determinism is due either to over-generation (cases where the grammar is not sufficiently constrained and allows for one tree to adjoin to another tree in several places) or to spurious derivation (distinct derivations with identical semantics). The few remaining cases that are linguistically correct are due to varying modifier positions and could be constrained by a sophisticated feature decorations in the elementary tree.

[4] This is not absolutely true as a tree identifier only reflects part of the compilation process. In practice, they are few exceptions though so that distinct trees whose tree identifiers are identical can be manually distinguished.

equivalent to the combination of other classes $C_1 \ldots C_n$ and consequently a tree identifier containing $C, C_1 \ldots C_n$ can be reduced to include either C or $C_1 \ldots C_n$.

2. Each literal l_i in the input is associated with a tree property set TP_i (i.e., the input we generate from is enriched with syntactic information)

3. During realisation, for each literal/tree property pair $\langle l_i : TP_i \rangle$ in the enriched input semantics, lexical selection is constrained to retrieve only those trees (i) whose semantics subsumes l_i and (ii) whose tree properties are TP_i

Since each literal is associated with a (simplified) tree identifier and each tree identifier uniquely identifies an elementary tree, realisation produces at most one realisation.

Examples 4a-4c illustrates the kind of constraints used by the realiser.

(4) a. l_j*:jean(j)/ProperName*
l_a*:aime(e,j,m)/[CanonicalNominalSubject, ActiveVerbForm, CanonicalNominalObject]*
l_m*:marie(m)/ProperName*
Jean aime Marie
* Jean est aimé de Marie

b. l_c*:le(c)/Det*
l_c*:chien(c)/Noun*
l_d*:dort(e1,c)/RelativeSubject*
l_r*:ronfle(e2,c)/CanonicalSubject*
Le chien qui dort ronfle
* Le chien qui ronfle dort

c. l_j*:jean(j)/ProperName*
l_p*:promise(e1,j,m,e2)/[CanonicalNominalSubject, ActiveVerbForm, CompletiveObject]*
l_m*:marie(m)/ProperName*
l_{e2}*:partir(e2,j)/InfinitivalVerb*
Jean promet à marie de partir
* Jean promet à marie qu'il partira

4.2 At least one realisation

For a realiser to be usable by a generation system, there must be some means to ensure that its input is satisfiable i.e., that it can be realised. How can this be done without actually carrying out realisation i.e., without checking that the input is satisfiable? Existing realisers indicate two types of answers to that dilemma.

A first possibility would be to draw on (Yang et al., 1991)'s proposal and compute the enriched input based on the traversal of a systemic network. More specifically, one possibility would be to consider a systemic network such as NIGEL, precompile all the functional features associated with each possible traversal of the network, map them onto the corresponding tree properties and use the resulting set of tree properties to ensure the satisfiability of the enriched input.

Another option would be to check the well formedness of the input at some level of the linguistic theory on which the realiser is based. Thus for instance, REALPRO assumes as input a well formed deep syntactic structure (DSyntS) as defined by Meaning Text Theory (MTT) and similarly, SURGE takes as input a functional description (FD) which in essence is an underspecified grammatical structure within the SURGE grammar. In both cases, there is no guarantee that the input be satisfiable since all the other levels of the linguistic theory must be verified for this to be true. In MTT, the DSyntS must first be mapped onto a surface syntactic structure and then successively onto the other levels of the theory while in SURGE, the input FD can be realised only if it provides consistent information for a complete top-down traversal of the grammar right down to the lexical level. In short, in both cases, the well formedness of the input can be checked with respect to some criteria (e.g., well formedness of a deep syntactic structure in MTT, well formedness of a FD in SURGE) but this well formedness does not guarantee satisfiability. Nonetheless this basic well formedness check is important as it provides some guidance as to what an acceptable input to the realiser should look like.

We adopt a similar strategy and resort to the notion of *polarity neutral input* to control the well formedness of the enriched input. The proposal draws on ideas from (Koller and Striegnitz, 2002; Gardent and Kow, 2005) and aims to determine whether for a given input (a set of TAG elementary trees whose semantics equate the input semantics), syntactic requirements and resources cancel out. More specifically, the aim is to determine whether given the input set of elementary trees, each substitution and each adjunction requirement is satisfied by exactly one elementary tree of the appropriate syntactic category and semantic index.

Roughly,[5] the technique consists in (automatically) associating with each elementary tree a polarity signature reflecting its substitution/adjunction requirements and resources and in computing the grand polarity of each possible combination of trees covering the input semantics. Each such combination whose total polarity is non-null is then filtered out (not considered for realisation) as it cannot possibly lead to a valid derivation (either a requirement cannot be satisfied or a resource cannot be used).

In the context of a generation system, polarity checking can be used to check the satisfiability of the input or more interestingly, to correct an ill formed input i.e., an input which can be detected as being unsatisfiable.

To check a given input, it suffices to compute its polarity count. If it is non-null, the input is unsatisfiable and should be revised. This is not very useful however, as the enriched input ensures determinism and thereby make realisation very easy, indeed almost as easy as polarity checking.

More interestingly, polarity checking can be used to suggest ways of fixing an ill formed input. In such a case, the enriched input is stripped of its control annotations, realisation proceeds on the basis of this simplified input and polarity checking is used to preselect all polarity neutral combinations of elementary trees. A closest match (i.e. the polarity neutral combination with the greatest number of control annotations in common with the ill formed input) to the ill formed input is then proposed as a probably satisfiable alternative.

5 Evaluation

To evaluate both the paraphrastic power of the realiser and the impact of the control annotations on non-determinism, we used a graduated test-suite which was built by (i) parsing a set of sentences, (ii) selecting the correct meaning representations from the parser output and (iii) generating from these meaning representations. The gradation in the test suite complexity was obtained by partitioning the input into sentences containing one, two or three finite verbs and by choosing cases allowing for different paraphrasing patterns. More specifically, the test suite includes cases involving the following types of paraphrases:

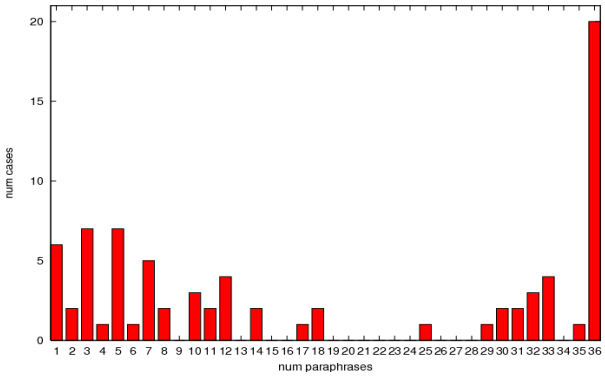
- Grammatical variations in the realisations of the arguments (cleft, cliticisation, question, relativisation, subject-inversion, etc.) or of the verb (active/passive, impersonal)
- Variations in the realisation of modifiers (e.g., relative clause vs adjective, predicative vs non-predicative adjective)
- Variations in the position of modifiers (e.g., pre- vs post-nominal adjective)
- Variations licensed by a morpho-derivational link (e.g., to arrive/arrival)

On a test set of 80 cases, the paraphrastic level varies between 1 and over 50 with an average of 18 paraphrases per input (taking 36 as upper cut off point in the paraphrases count). Figure 5 gives a more detailed description of the distribution of the paraphrastic variation. In essence, 42% of the sentences with one finite verb accept 1 to 3 paraphrases (cases of intransitive verbs), 44% accept 4 to 28 paraphrases (verbs of arity 2) and 13% yield more than 29 paraphrases (ditransitives). For sentences containing two finite verbs, the ratio is 5% for 1 to 3 paraphrases, 36% for 4 to 14 paraphrases and 59% for more than 14 paraphrases. Finally, sentences containing 3 finite verbs all accept more than 29 paraphrases.

Two things are worth noting here. First, the paraphrase figures might seem low wrt to e.g., work by (Velldal and Oepen, 2006) which mentions several thousand outputs for one given input and an average number of realisations per input varying between 85.7 and 102.2. Admittedly, the French grammar we are using has a much more limited coverage than the ERG (the grammar used by (Velldal and Oepen, 2006)) and it is possible that its paraphrastic power is lower. However, the counts we give only take into account *valid paraphrases of the input*. In other words, overgeneration and spurious derivations are excluded from the toll. This does not seem to be the case in (Velldal and Oepen, 2006)'s approach where the count seems to include *all* sentences associated by the grammar with the input semantics.

Second, although the test set may seem small it is important to keep in mind that it represents 80 inputs

[5] Lack of space prevents us from giving much details here. We refer the reader to (Koller and Striegnitz, 2002; Gardent and Kow, 2005) for more details.



with distinct grammatical and paraphrastic properties. In effect, these 80 test cases yields 1 528 distinct well-formed sentences. This figure compares favourably with the size of the largest regression test suite used by a symbolic NLG realiser namely, the SURGE test suite which contains 500 input each corresponding to a single sentence. It also compares reasonably with other more recent evaluations (Callaway, 2003; Langkilde-Geary, 2002) which derive their input data from the Penn Treebank by transforming each sentence tree into a format suitable for the realiser (Callaway, 2003). For these approaches, the test set size varies between roughly 1 000 and almost 3 000 sentences. But again, it is worth stressing that these evaluations aim at assessing coverage and correctness (does the realiser find the sentence used to derive the input by parsing it?) rather than the paraphrastic power of the grammar. They fail to provide a systematic assessment of how many distinct grammatical paraphrases are associated with each given input.

To verify the claim that tree properties can be used to ensure determinism (cf. footnote 4), we started by eliminating from the output all ill-formed sentences. We then automatically associated each well-formed output with its set of tree properties. Finally, for each input semantics, we did a systematic pairwise comparison of the tree property sets associated with the input realisations and we checked whether for any given input, there were two (or more) distinct paraphrases whose tree properties were the same. We found that such cases represented slightly over 2% of the total number of (input,realisations) pairs. Closer investigation of the faulty data indicates two main reasons for non-determinism namely, trees with alternating order of arguments and derivations with distinct modifier adjunctions. Both cases can be handled by modifying the grammar in such a way that those differences are reflected in the tree properties.

6 Related work

The approach presented here combines a reversible grammar realiser with a symbolic approach to paraphrase selection. We now compare it to existing surfaces realisers.

NLG geared realisers. Prominent general purpose NLG geared realisers include REALPRO, SURGE, KPML, NITROGEN and HALOGEN. Furthermore, HALOGEN has been shown to achieve broad coverage and high quality output on a set of 2 400 input automatically derived from the Penn treebank.

The main difference between these and the present approach is that our approach is based on a reversible grammar whilst NLG geared realisers are not. This has several important consequences.

First, it means that one and *the same grammar and lexicon can be used both for parsing and for generation*. Given the complexity involved in developing such resources, this is an important feature.

Second, as demonstrated in the Redwood Lingo Treebank, reversibility makes it easy to *rapidly create very large evaluation suites*: it suffices to parse a set of sentences and select from the parser output the correct semantics. In contrast, NLG geared realisers either work on evaluation sets of restricted size (500 input for SURGE, 210 for KPML) or require the time expensive implementation of a preprocessor transforming e.g., Penn Treebank trees into a format suitable for the realisers. For instance, (Callaway, 2003) reports that the implementation of such a processor for SURGE was the most time consuming part of the evaluation with the resulting component containing 4000 lines of code and 900 rules.

Third, a reversible grammar can be exploited to *support not only realisation but also its reverse, namely semantic construction*. Indeed, reversibility is ensured through a compositional semantics that is, through a tight coupling between syntax and semantics. In contrast, NLG geared realisers often have to reconstruct this association in rather ad hoc ways. Thus for instance, (Yang et al., 1991) resorts to ad

hoc "mapping tables" to associate substitution nodes with semantic indices and "fr-nodes" to constrain adjunction to the correct nodes. More generally, the lack of a clearly defined compositional semantics in NLG geared realisers makes it difficult to see how the grammar they use could be exploited to also support semantic construction.

Fourth, the *grammar can be used both to generate and to detect paraphrases*. It could be used for instance, in combination with the parser and the semantic construction module described in (Gardent and Parmentier, 2005), to support textual entailment recognition or answer detection in question answering.

Reversible realisers. The realiser presented here differs in mainly two ways from existing reversible realisers such as (White, 2004)'s CCG system or the HPSG ERG based realiser (Carroll and Oepen, 2005).

First, it permits a symbolic selection of the output paraphrase. In contrast, existing reversible realisers use statistical information to select from the produced output the most plausible paraphrase.

Second, particular attention has been paid to the treatment of paraphrases in the grammar. Recall that TAG elementary trees are grouped into families and further, that the specific TAG we use is compiled from a highly factorised description. We rely on these features to associate one and the same semantic to large sets of trees denoting semantically equivalent but syntactically distinct configurations (cf. (Gardent, 2006)).

7 Conclusion

The realiser presented here, GENI, exploits a grammar which is produced semi-automatically by compiling a high level grammar description into a Tree Adjoining Grammar. We have argued that a side-effect of this compilation process – namely, the association with each elementary tree of a set of tree properties – can be used to constrain the realiser output. The resulting system combines the advantages of two orthogonal approaches. From the reversible approach, it takes the reusability, the ability to rapidly create very large test suites and the capacity to both generate and detect paraphrases. From the NLG geared paradigm, it takes the ability to symbolically constrain the realiser output to a given generation context.

GENI is free (GPL) software and is available at `http://trac.loria.fr/~geni`.

References

Charles B. Callaway. 2003. Evaluating coverage for large symbolic NLG grammars. In *18th IJCAI*, pages 811–817, Aug.

J. Carroll and S. Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. *2nd IJCNLP*.

B. Crabbé and D. Duchier. 2004. Metagrammar redux. In *CSLP*, Copenhagen.

M. Elhadad and J. Robin. 1999. SURGE: a comprehensive plug-in syntactic realization component for text generation. *Computational Linguistics*.

C. Gardent and L. Kallmeyer. 2003. Semantic construction in FTAG. In *10th EACL*, Budapest, Hungary.

C. Gardent and E. Kow. 2005. Generating and selecting grammatical paraphrases. *ENLG*, Aug.

C. Gardent and Y. Parmentier. 2005. Large scale semantic construction for Tree Adjoining Grammars. *LACL05*.

C. Gardent. 2006. Integration d'une dimension semantique dans les grammaires d'arbres adjoints. *TALN*.

M. Kay. 1996. Chart Generation. In *34th ACL*, pages 200–204, Santa Cruz, California.

A. Koller and K. Striegnitz. 2002. Generation as dependency parsing. In *40th ACL*, Philadelphia.

I. Langkilde-Geary. 2002. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proceedings of the INLG*.

B. Lavoie and O. Rambow. 1997. RealPro–a fast, portable sentence realizer. *ANLP'97*.

C. Matthiessen and J.A. Bateman. 1991. *Text generation and systemic-functional linguistics: experiences from English and Japanese*. Frances Pinter Publishers and St. Martin's Press, London and New York.

I.A. Mel'cuk. 1988. *Dependency Syntax: Theorie and Practice*. State University Press of New York.

Erik Velldal and Stephan Oepen. 2006. Statistical ranking in tactical generation. In *EMNLP*, Sydney, Australia.

K. Vijay-Shanker and AK Joshi. 1988. Feature Structures Based Tree Adjoining Grammars. *Proceedings of the 12th conference on Computational linguistics*, 55:v2.

M. White. 2004. Reining in CCG chart realization. In *INLG*, pages 182–191.

G. Yang, K. McKoy, and K. Vijay-Shanker. 1991. From functional specification to syntactic structure. *Computational Intelligence*, 7:207–219.

Sentence generation as a planning problem

Alexander Koller
Center for Computational Learning Systems
Columbia University
koller@cs.columbia.edu

Matthew Stone
Computer Science
Rutgers University
Matthew.Stone@rutgers.edu

Abstract

We translate sentence generation from TAG grammars with semantic and pragmatic information into a planning problem by encoding the contribution of each word declaratively and explicitly. This allows us to exploit the performance of off-the-shelf planners. It also opens up new perspectives on referring expression generation and the relationship between language and action.

1 Introduction

Systems that produce natural language must synthesize the primitives of linguistic structure into well-formed utterances that make desired contributions to discourse. This is fundamentally a planning problem: Each linguistic primitive makes certain contributions while potentially introducing new goals. In this paper, we make this perspective explicit by translating the sentence generation problem of TAG grammars with semantic and pragmatic information into a planning problem stated in the widely used Planning Domain Definition Language (PDDL, McDermott (2000)). The encoding provides a clean separation between computation and linguistic modelling and is open to future extensions. It also allows us to benefit from the past and ongoing advances in the performance of off-the-shelf planners (Blum and Furst, 1997; Kautz and Selman, 1998; Hoffmann and Nebel, 2001).

While there have been previous systems that encode generation as planning (Cohen and Perrault, 1979; Appelt, 1985; Heeman and Hirst, 1995), our approach is distinguished from these systems by its focus on the grammatically specified contributions of each individual word (and the TAG tree it anchors) to syntax, semantics, and local pragmatics (Hobbs et al., 1993). For example, words directly achieve content goals by adding a corresponding semantic primitive to the conversational record. We deliberately avoid reasoning about utterances as coordinated rational behavior, as earlier systems did; this allows us to get by with a much simpler logic.

The problem we solve encompasses the generation of referring expressions (REs) as a special case. Unlike some approaches (Dale and Reiter, 1995; Heeman and Hirst, 1995), we do not have to distinguish between generating NPs and expressions of other syntactic categories. We develop a new perspective on the lifecycle of a distractor, which allows us to generate more succinct REs by taking the rest of the utterance into account. More generally, we do not split the process of sentence generation into two separate steps of sentence planning and realization, as most other systems do, but solve the joint problem in a single integrated step. This can potentially allow us to generate higher-quality sentences. We share these advantages with systems such as SPUD (Stone et al., 2003).

Crucially, however, our approach describes the dynamics of interpretation explicitly and declaratively. We do not need to assume extra machinery beyond the encoding of words as PDDL planning operators; for example, our planning operators give a self-contained description of how each individual word contributes to resolving references. This makes our encoding more direct and transparent than those in work like Thomason and Hobbs (1997) and Stone et al. (2003).

We present our encoding in a sequence of steps, each of which adds more linguistic information to

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 336–343,
Prague, Czech Republic, June 2007.

the planning operators. After a brief review of LTAG and PDDL, we first focus on syntax alone and show how to cast the problem of generating grammatically well-formed LTAG trees as a planning problem in Section 2. In Section 3, we add semantics to the elementary trees and add goals to communicate specific content (this corresponds to surface realization). We complete the account by modeling referring expressions and go through an example. Finally, we assess the practical efficiency of our approach and discuss future work in Section 4.

2 Grammaticality as planning

We start by reviewing the LTAG grammar formalism and giving an intuition of how LTAG generation is planning. We then add semantic roles to the LTAG elementary trees in order to distinguish different substitution nodes. Finally, we review the PDDL planning specification language and show how LTAG grammaticality can be encoded as a PDDL problem and how we can reconstruct an LTAG derivation from the plan.

2.1 Tree-adjoining grammars

The grammar formalism we use here is that of *lexicalized tree-adjoining grammars* (LTAG; Joshi and Schabes (1997)). An LTAG grammar consists of a finite set of lexicalized *elementary trees* as shown in Fig. 1a. Each elementary tree contains exactly one *anchor* node, which is labelled by a word. Elementary trees can contain *substitution nodes*, which are marked by down arrows (↓). Those elementary trees that are *auxiliary trees* also contain exactly one *foot node*, which is marked with an asterisk (∗). Trees that are not auxiliary trees are called *initial trees*.

Elementary trees can be combined by *substitution* and *adjunction* to form larger trees. Substitution is the operation of replacing a substitution node of some tree by another initial tree with the same root label. Adjunction is the operation of splicing an auxiliary tree into some node v of a tree, in such a way that the root of the auxiliary tree becomes the child of v's parent, and the foot node becomes the parent of v's children. If a node carries a *null adjunction constraint* (indicated by no-adjoin), no adjunction is allowed at this node; if it carries an *obligatory adjunction constraint* (indicated by adjoin!), an auxiliary tree *must* be adjoined there.

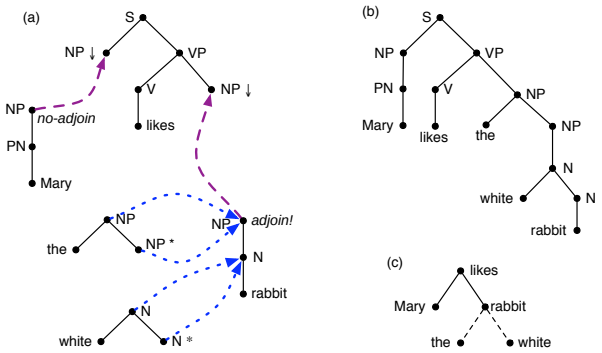


Figure 1: Building a derived (b) and a derivation tree (c) by combining elementary trees (a).

In Fig. 1a, we have combined some elementary trees by substitution (indicated by the dashed/magenta arrows) and adjunction (dotted/blue arrows). The result of these operations is the *derived tree* in Fig. 1b. The *derivation tree* in Fig. 1c represents the tree combination operations we used by having one node per elementary tree and drawing a solid edge if we combined the two trees by substitution, and a dashed edge for adjunctions.

2.2 The basic idea

Consider the process of constructing a derivation tree top-down. To build the tree in Fig. 1c, say, we start with the empty derivation tree and an obligation to generate an expression of category S. We satisfy this obligation by adding the tree for "likes" as the root of the derivation; but in doing so, we have introduced new unfilled substitution nodes of category NP, i.e. the derivation tree is not complete. We use the NP tree for "Mary" to fill one substitution node and the NP tree for "rabbit" to fill the other. This fills both substitution nodes, but the "rabbit" tree introduces an obligatory adjunction constraint, which we must satisfy by adjoining the auxiliary tree for "the". We now have a grammatical derivation tree, but we are free to continue by adding more auxiliary trees, such as the one for "white".

As we have just presented it, the generation of derivation trees is essentially a planning problem. A planning problem involves *states* and *actions* that can move from one state to another. The task is to find a sequence of actions that moves us from the

initial state to a state that satisfies all the *goals*. In our case, the states are defined by the unfilled substitution nodes, the unsatisfied obligatory adjunction constraints, and the nodes that are available for adjunction in some (possibly incomplete) derivation tree. Each action adds a single elementary tree to the derivation, removing some of these "open nodes" while introducing new ones. The initial state is associated with the empty derivation tree and a requirement to generate an expression for the given root category. The goal is for the current derivation tree to be grammatically complete.

2.3 Semantic roles

Formalizing this intuition requires unique names for each node in the derived tree. Such names are necessary to distinguish the different open substitution nodes that still need to be filled, or the different available adjunction sites; in the example, the planner needed to be aware that "likes" introduces *two* separate NP substitution nodes to fill.

There are many ways to assign these names. One that works particularly well in the context of PDDL (as we will see below) is to assume that each node in an elementary tree, except for ones with null adjunction constraints, is marked with a *semantic role*, and that all substitution nodes are marked with different roles. Nothing hinges on the particular role inventory; here we assume an inventory including the roles ag for "agent" and pat for "patient". We also assume one special role self, which must be used for the root of each elementary tree and must never be used for substitution nodes.

We can now assign a unique name to every substitution node in a derived tree by assigning arbitrary but distinct *indices* to each use of an elementary tree, and giving the substitution node with role r in the elementary tree with index i the *identity* $i.r$. In the example, let's say the "likes" tree has index 1 and the semantic roles for the substitution nodes were ag and pat, respectively. The planner action that adds this tree would then require substitution of one NP with identity 1.ag and another NP with identity 1.pat; the "Mary" tree would satisfy the first requirement and the "rabbit" tree the second. If we assume that no elementary tree contains two internal nodes with the same category and role, we can refer to adjunction opportunities in a similar way.

Action `S-likes-1`(u). Precond: $\mathsf{subst}(\mathrm{S},u),\mathsf{step}(1)$
Effect: $\neg\mathsf{subst}(\mathrm{S},u),\mathsf{subst}(\mathrm{NP},1.\mathsf{ag}),$
$\mathsf{subst}(\mathrm{NP},1.\mathsf{pat}),\neg\mathsf{step}(1),\mathsf{step}(2)$

Action `NP-Mary-2`(u). Precond: $\mathsf{subst}(\mathrm{NP},u),\mathsf{step}(2)$
Effect: $\neg\mathsf{subst}(\mathrm{NP},u),\neg\mathsf{step}(2),\mathsf{step}(3)$

Action `NP-rabbit-3`(u). Precond: $\mathsf{subst}(\mathrm{NP},u),\mathsf{step}(3)$
Effect: $\neg\mathsf{subst}(\mathrm{NP},u),\mathsf{canadjoin}(\mathrm{NP},u),$
$\mathsf{mustadjoin}(\mathrm{NP},u),\neg\mathsf{step}(3),\mathsf{step}(4)$

Action `NP-the-4`(u). Precond: $\mathsf{canadjoin}(\mathrm{NP},u),\mathsf{step}(4)$
Effect: $\neg\mathsf{mustadjoin}(\mathrm{NP},u),\neg\mathsf{step}(4),\mathsf{step}(5)$

Figure 2: Some actions for the grammar in Fig. 1.

2.4 Encoding in PDDL

Now we are ready to encode the problem of generating grammatical LTAG derivation trees into PDDL. PDDL (McDermott, 2000) is the standard input language for modern planning systems. It is based on the well-known STRIPS language (Fikes and Nilsson, 1971). In this paradigm, a planning state is defined as a finite set of ground atoms of predicate logic that are true in this state; all other atoms are assumed to be false. Actions have a number of *parameters*, as well as a *precondition* and *effect*, both of which are logical formulas. When a planner tries to apply an action, it will first create an action *instance* by binding all parameters to constants from the domain. It must then verify that the precondition of the action instance is satisfied in the current state. If so, the action can be applied, in which case the effect is processed in order to change the state. In STRIPS, the precondition and effect both had to be conjunctions of atoms or negated atoms; positive effects are interpreted as making the atom true in the new state, and negative ones as making it false. PDDL permits numerous extensions to the formulas that can be used as preconditions and effects.

Each action in our planning problem encodes the effect of adding some elementary tree to the derivation tree. An initial tree with root category A translates to an action with a parameter u for the identity of the node that the current tree is substituted into. The action carries the precondition $\mathsf{subst}(A,u)$, and so can only be applied if u is an open substitution node in the current derivation with the correct category A. Auxiliary trees are analogous, but carry the precondition $\mathsf{canadjoin}(A,u)$. The effect of an initial tree is to remove the subst condition from the planning state (to record that the substitu-

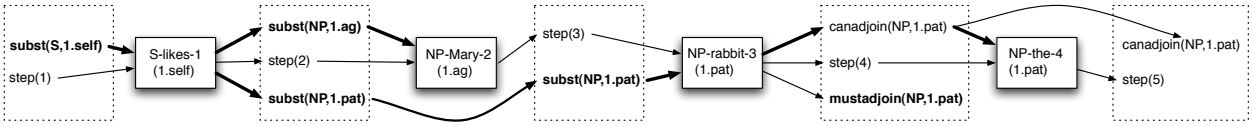


Figure 3: A plan for the actions in Fig. 2.

tion node u is now filled); an auxiliary tree has an effect $\neg\mathsf{mustadjoin}(A,u)$ to indicate that any obligatory adjunction constraint is satisfied but leaves the canadjoin condition in place to allow multiple adjunctions into the same node. In both cases, effects add subst, canadjoin and mustadjoin atoms representing the substitution nodes and adjunction sites that are introduced by the new elementary tree.

One remaining complication is that an action must assign new identities to the nodes it introduces; thus it must have access to a tree index that was not used in the derivation tree so far. We use the number of the current plan step as the index. We add an atom $\mathsf{step}(1)$ to the initial state of the planning problem, and we introduce k different copies of the actions for each elementary tree, where k is some upper limit on the plan size. These actions are identical, except that the i-th copy has an extra precondition $\mathsf{step}(i)$ and effects $\neg\mathsf{step}(i)$ and $\mathsf{step}(i+1)$. It is no restriction to assume an upper limit on the plan size, as most modern planners search for plans smaller than a given maximum length anyway.

Fig. 2 shows some of the actions into which the grammar in Fig. 1 translates. We display only one copy of each action and have left out most of the canadjoin effects. In addition, we use an initial state containing the atoms $\mathsf{subst}(S,1.\mathsf{self})$ and $\mathsf{step}(1)$ and a final state consisting of the following goal:

$$\forall A,u.\neg\mathsf{subst}(A,u) \wedge \forall A,u.\neg\mathsf{mustadjoin}(A,u).$$

We can then send the actions and the initial state and goal specifications to any off-the-shelf planner and obtain the plan in Fig. 3. The straight arrows in the picture link the actions to their preconditions and (positive) effects; the curved arrows indicate atoms that carry over from one state to the next without being changed by the action. Atoms are printed in boldface iff they contradict the goal.

This plan can be read as a derivation tree that has one node for each action instance in the plan, and an edge from node u to node v if u establishes a subst or canadjoin fact that is a precondition of v. These causal links are drawn as bold edges in Fig. 3. The mapping is unique for substitution edges because subst atoms are removed by every action that has them as their precondition. There may be multiple action instances in the plan that introduce the same atom $\mathsf{canadjoin}(A,u)$. In this case, we can freely choose one of these instances as the parent.

3 Sentence generation as planning

Now we extend this encoding to deal with semantics and referring expressions.

3.1 Communicative goals

In order to use the planner as a *surface realization* algorithm for TAG along the lines of Koller and Striegnitz (2002), we attach *semantic content* to each elementary tree and require that the sentence achieves a certain *communicative goal*. We also use a *knowledge base* that specifies the speaker's knowledge, and require that we can only use trees that express information in this knowledge base.

We follow Stone et al. (2003) in formalizing the semantic content of a lexicalized elementary tree t as a finite set of atoms; but unlike in earlier approaches, we use the semantic roles in t as the arguments of these atoms. For instance, the semantic content of the "likes" tree in Fig. 1 is $\{\mathsf{like}(\mathsf{self},\mathsf{ag},\mathsf{pat})\}$ (see also the semcon entries in Fig. 4). The knowledge base is some finite set of ground atoms; in the example, it could contain such entries as $\mathsf{like}(e,m,r)$ and $\mathsf{rabbit}(r)$. Finally, the communicative goal is some subset of the knowledge base, such as $\{\mathsf{like}(e,m,r)\}$.

We implement unsatisfied communicative goals as flaws that the plan must remedy. To this end, we add an atom $\mathsf{cg}(P,a_1,\ldots,a_n)$ for each element $P(a_1,\ldots,a_n)$ of the communicative goal to the initial state, and we add a corresponding conjunct $\forall P,x_1,\ldots,x_n.\neg\mathsf{cg}(P,x_1,\ldots,x_n)$ to the goal. In addition, we add an atom $\mathsf{skb}(P,a_1,\ldots,a_n)$ to the initial state for each element $P(a_1,\ldots,a_n)$ of the (speaker's) knowledge base.

We then add parameters $x_1, \ldots, x_n$ to each action with n semantic roles (including self). These new parameters are intended to be bound to individual constants in the knowledge base by the planner. For each elementary tree t and possible step index i, we establish the relationship between these parameters and the roles in two steps. First we fix a function id that maps the semantic roles of t to node identities. It maps self to u and each other role r to $i.r$. Second, we fix a function ref that maps the outputs of id bijectively to the parameters $x_1, \ldots, x_n$, in such a way that $\mathsf{ref}(u) = x_1$.

We can then capture the contribution of the i-th action for t to the communicative goal by giving it an effect $\neg\mathsf{cg}(P, \mathsf{ref}(\mathsf{id}(r_1)), \ldots, \mathsf{ref}(\mathsf{id}(r_n)))$ for each element $P(r_1, \ldots, r_n)$ of the elementary tree's semantic content. We restrict ourselves to only expressing true statements by giving the action a precondition $\mathsf{skb}(P, \mathsf{ref}(\mathsf{id}(r_1)), \ldots, \mathsf{ref}(\mathsf{id}(r_n)))$ for each element of the semantic content.

In order to keep track of the connection between node identities and individuals for future reference, each action gets an effect $\mathsf{referent}(\mathsf{id}(r), \mathsf{ref}(\mathsf{id}(r)))$ for each semantic role r except self. We enforce the connection between u and x_1 by adding a precondition $\mathsf{referent}(u, x_1)$.

In the example, the most interesting action in this respect is the one for the elementary tree for "likes". This action looks as follows:

Action `S-likes-1`(u, x_1, x_2, x_3)**.**

Precond: $\mathsf{subst}(\mathrm{S}, u), \mathsf{step}(1), \mathsf{referent}(u, x_1),$
$\mathsf{skb}(\mathsf{like}, x_1, x_2, x_3)$
Effect: $\neg\mathsf{subst}(\mathrm{S}, u), \mathsf{subst}(\mathrm{NP}, 1.\mathsf{ag}), \mathsf{subst}(\mathrm{NP}, 1.\mathsf{pat}),$
$\neg\mathsf{step}(1), \mathsf{step}(2),$
$\mathsf{referent}(1.\mathsf{ag}, x_2), \mathsf{referent}(1.\mathsf{pat}, x_3),$
$\neg\mathsf{cg}(\mathsf{like}, x_1, x_2, x_3)$

We can run a planner and interpret the plan as above; the main difference is that complete plans not only correspond to grammatical derivation trees, but also express all communicative goals. Notice that this encoding models some aspects of lexical choice: The semantic content sets of the elementary trees need not be singletons, and so there may be multiple ways of partitioning the communicative goal into the content sets of various elementary trees.

3.2 Referring expressions

Finally, we extend the system to deal with the generation of referring expressions. While this problem is typically taken to require the generation of a noun phrase that refers uniquely to some individual, we don't need to make any assumptions about the syntactic category here. Moreover, we consider the problem in the wider context of generating referring expressions within a sentence, which can allow us to generate more succinct expressions.

Because a referring expression must allow the *hearer* to identify the intended referent uniquely, we keep track of the hearer's knowledge base separately. We use atoms $\mathsf{hkb}(P, a_1, \ldots, a_n)$, as with skb above. In addition, we assume *pragmatic* information of the form $\mathsf{pkb}(P, a_1, \ldots, a_n)$. The three pragmatic predicates that we will use here are hearer-new, indicating that the hearer does not know about the existence of an individual and can't infer it (Stone et al., 2003), hearer-old for the opposite, and contextset. The *context set* of an intended referent is the set of all individuals that the hearer might possibly confuse it with (DeVault et al., 2004). It is empty for hearer-new individuals. To say that b is in a's context set, we put the atom $\mathsf{pkb}(\mathsf{contextset}, a, b)$ into the initial state.

In addition to the semantic content, we equip every elementary tree in the grammar with a *semantic requirement* and a *pragmatic condition* (Stone et al., 2003). The semantic requirement is a set of atoms spelling out presuppositions of an elementary tree that can help the hearer identify what its arguments refer to. For instance, "likes" has the selectional restriction that its agent must be animate; thus the hearer will not consider inanimate individuals as distractors for the referring expression in agent position. The pragmatic condition is a set of atoms over the predicates in the pragmatic knowledge base.

In our setting, every substitution node that is introduced during the derivation introduces a new referring expression. This means that we can distinguish the referring expressions by the identity of the substitution node that introduced them. For each referring expression u (where u is a node identity), we keep track of the *distractors* in atoms of the form $\mathsf{distractor}(u, x)$. The presence of an atom $\mathsf{distractor}(u, a)$ in some planning state represents the fact that the current derivation tree is not yet informative enough to allow the hearer to identify the intended referent for u uniquely; a is another individual that is not the intended referent,

but consistent with the partial referring expression we have constructed so far. We enforce uniqueness of all referring expressions by adding the conjunct $\forall u,x \neg \mathsf{distractor}(u,x)$ to the planning goal.

Now whenever an action introduces a new substitution node u, it will also introduce some distractor atoms to record the initial distractors for the referring expression at u. An individual a is in the initial distractor set for the substitution node with role r if (a) it is not the intended referent, (b) it is in the context set of the intended referent, and (c) there is a choice of individuals for the other parameters of the action that satisfies the semantic requirement together with a. This is expressed by adding the following effect for each substitution node; the conjunction is over the elements $P(r_1,\ldots,r_n)$ of the semantic requirement, and there is one universal quantifier for y and for each parameter x_j of the action except for $\mathsf{ref}(\mathsf{id}(r))$.

$$\begin{array}{l}\forall y,x_1,\ldots,x_n \\ \quad (y \neq \mathsf{ref}(\mathsf{id}(r)) \wedge \mathsf{pkb}(\mathsf{contextset},\mathsf{ref}(\mathsf{id}(r)),y) \wedge \\ \quad \bigwedge \mathsf{hkb}(P,\mathsf{ref}(\mathsf{id}(r_1)),\ldots,\mathsf{ref}(\mathsf{id}(r_n)))[y/\mathsf{ref}(\mathsf{id}(r))]) \\ \quad \rightarrow \mathsf{distractor}(\mathsf{id}(r),y)\end{array}$$

On the other hand, a distractor a for a referring expression introduced at u is removed when we substitute or adjoin an elementary tree into u which rules a out. For instance, the elementary tree for "rabbit" will remove all non-rabbits from the distractor set of the substitution node into which it is substituted. We achieve this by adding the following effect to each action; here the conjunction is over all elements of the semantic content.

$$\begin{array}{l}\forall y.(\neg \bigwedge \mathsf{hkb}(P,\mathsf{ref}(\mathsf{id}(r_1)),\ldots,\mathsf{ref}(\mathsf{id}(r_n))))[y/x_1] \\ \quad \rightarrow \neg \mathsf{distractor}(u,y),\end{array}$$

Finally, each action gets its pragmatic condition as a precondition.

3.3 The example

By way of example, Fig. 5 shows the full versions of the actions from Fig. 2, for the extended grammar in Fig. 4. Let's say that the hearer knows about two rabbits r (which is white) and r' (which is not), about a person m with the name Mary, and about an event e, and that the context set of r is $\{r,r',m,e\}$. Let's also say that our communicative goal is $\{\mathsf{like}(e,m,r)\}$. In this case, the first action instance in Fig. 3, `S-likes-1`$(1.\mathsf{self},e,m,r)$, introduces a substitution node with identity 1.pat. The

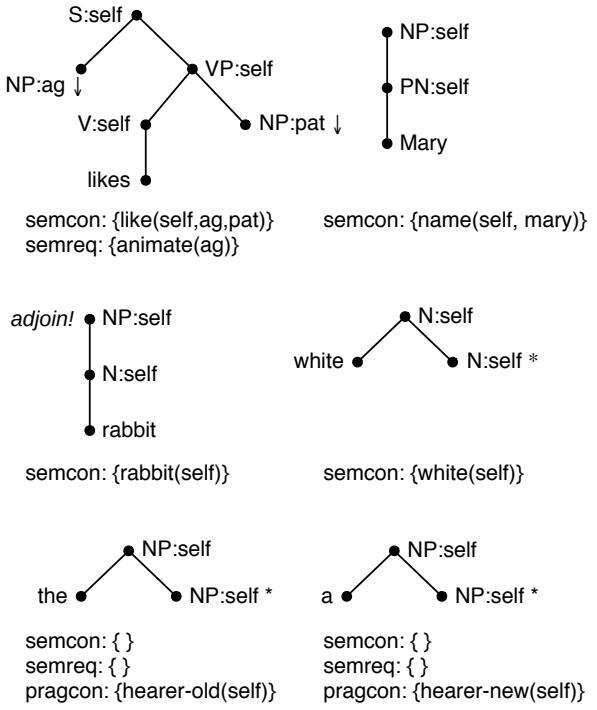


Figure 4: The extended example grammar.

initial distractor set of this node is $\{r',m\}$ – the set of all individuals in r's context set except for inanimate objects (which violate the semantic requirement) and r itself. The `NP-rabbit-3` action removes m from the distractor set, but at the end of the plan in Fig. 3, r' is still a distractor, i.e. we have not reached a goal state. We can complete the plan by performing a final action `NP-white-5`$(1.\mathsf{pat},r)$, which will remove this distractor and achieve the planning goal. We can still reconstruct a derivation tree from the complete plan literally as described in Section 2.

Now let's say that the hearer did not know about the existence of the individual r before the utterance we are generating. We model this by marking r as hearer-new in the pragmatic knowledge base and assigning it an empty context set. In this case, the referring expression 1.pat would be initialized with an empty distractor set. This entitles us to use the action `NP-a-4` and generate the four-step plan corresponding to the sentence "Mary likes a rabbit."

4 Discussion and future work

In conclusion, let's look in more detail at computational issues and the role of mutually constraining referring expressions.

Action `S-likes-1`(u,x_1,x_2,x_3)**.**
Precond: $\mathsf{referent}(u,x_1),\mathsf{skb}(\mathsf{like},x_1,x_2,x_3),\mathsf{subst}(\mathrm{S},u),\mathsf{step}(1)$
Effect: $\neg\mathsf{cg}(\mathsf{like},x_1,x_2,x_3),\neg\mathsf{subst}(\mathrm{S},u),\neg\mathsf{step}(1),\mathsf{step}(2),\mathsf{subst}(\mathrm{NP},1.\mathsf{ag}),\mathsf{subst}(\mathrm{NP},1.\mathsf{pat}),$
$\forall y.\neg\mathsf{hkb}(\mathsf{like},y,x_2,x_3)\rightarrow\neg\mathsf{distractor}(u,y),$
$\forall y,x_1,x_3.x_2\neq y\wedge\mathsf{pkb}(\mathsf{contextset},x_2,y)\wedge\mathsf{animate}(y)\rightarrow\mathsf{distractor}(1.\mathsf{ag},y),$
$\forall y,x_1,x_2.x_3\neq y\wedge\mathsf{pkb}(\mathsf{contextset},x_3,y)\rightarrow\mathsf{distractor}(1.\mathsf{pat},y)$

Action `NP-Mary-2`(u,x_1)**.**
Precond: $\mathsf{referent}(u,x_1),\mathsf{skb}(\mathsf{name},x_1,\mathsf{mary}),$
$\mathsf{subst}(\mathrm{NP},u),\mathsf{step}(2)$
Effect: $\neg\mathsf{cg}(\mathsf{name},x_1,\mathsf{mary}),\neg\mathsf{subst}(\mathrm{NP},u),$
$\neg\mathsf{step}(2),\mathsf{step}(3),$
$\forall y.\neg\mathsf{hkb}(\mathsf{name},y,\mathsf{mary})\rightarrow\neg\mathsf{distractor}(u,y)$

Action `NP-rabbit-3`(u,x_1)**.**
Precond: $\mathsf{referent}(u,x_1),\mathsf{skb}(\mathsf{rabbit},x_1),$
$\mathsf{subst}(\mathrm{N},u),\mathsf{step}(3)$
Effect: $\neg\mathsf{cg}(\mathsf{rabbit},x_1),\neg\mathsf{subst}(\mathrm{N},u),\neg\mathsf{step}(3),\mathsf{step}(4),$
$\mathsf{canadjoin}(\mathrm{NP},u),\mathsf{mustadjoin}(\mathrm{NP},u),$
$\forall y.\neg\mathsf{hkb}(\mathsf{rabbit},y)\rightarrow\neg\mathsf{distractor}(u,y)$

Action `NP-the-4`(u,x_1)**.**
Precond: $\mathsf{referent}(u,x_1),\mathsf{canadjoin}(\mathrm{NP},u),\mathsf{step}(4),$
$\mathsf{pkb}(\mathsf{hearer\text{-}old},x_1)$
Effect: $\neg\mathsf{mustadjoin}(\mathrm{NP},u),\neg\mathsf{step}(4),\mathsf{step}(5)$

Action `NP-a-4`(u,x_1)**.**
Precond: $\mathsf{referent}(u,x_1),\mathsf{canadjoin}(\mathrm{NP},u),\mathsf{step}(4),$
$\mathsf{pkb}(\mathsf{hearer\text{-}new},x_1)$
Effect: $\neg\mathsf{mustadjoin}(\mathrm{NP},u),\neg\mathsf{step}(4),\mathsf{step}(5)$

Action `NP-white-5`(u,x_1)**.**
Precond: $\mathsf{referent}(u,x_1),\mathsf{skb}(\mathsf{white},x_1),\mathsf{canadjoin}(\mathrm{NP},u),\mathsf{step}(5)$
Effect: $\neg\mathsf{cg}(\mathsf{white},x_1),\neg\mathsf{mustadjoin}(\mathrm{NP},u),\neg\mathsf{step}(5),\mathsf{step}(6),$
$\forall y.\neg\mathsf{hkb}(\mathsf{white},y)\rightarrow\neg\mathsf{distractor}(u,y)$

Figure 5: Some of the actions corresponding to the grammar in Fig. 4.

4.1 Computational issues

We lack the space to present the formal definition of the sentence generation problem we encode into PDDL. However, this problem is NP-complete, by reduction of Hamiltonian Cycle – unsurprisingly, given that it encompasses realization, and the very similar realization problem in Koller and Striegnitz (2002) is NP-hard. So any algorithm for our problem must be prepared for exponential runtimes.

We have implemented the translation described in this paper and experimented with a number of different grammars, knowledge bases, and planners. The FF planner (Hoffmann and Nebel, 2001) can compute the plans in Section 3.3 in under 100 ms using the grammar in Fig. 4. If we add 10 more lexicon entries to the grammar, the runtime grows to 190 ms; and for 20 more entries, to 360 ms. The runtime also grows with the plan length: It takes 410 ms to generate a sentence "Mary likes the Adj ... Adj rabbit" with four adjectives and 890 ms for six adjectives, corresponding to a plan length of 10. We compared these results against a planning-based reimplementation of SPUD's greedy search heuristic (Stone et al., 2003). This system is faster than FF for small inputs (360 ms for four adjectives), but becomes slower as inputs grow larger (1000 ms for six adjectives); but notice that while FF is also a heuristic planner, it is guaranteed to find a solution if one exists, unlike SPUD.

Planners have made tremendous progress in efficiency in the past decade, and by encoding sentence generation as a planning problem, we are set to profit from any future improvements; it is an advantage of the planning approach that we can compare very different search strategies like FF's and SPUD's in the same framework. However, our PDDL problems are challenging for modern planners because most planners start by computing all instances of atoms and actions. In our experiments, FF generally spent only about 10% of the runtime on search and the rest on computing the instances; that is, there is a lot of room for optimization. For larger grammars and knowledge bases, the number of instances can easily grow into the billions. In future work, we will therefore collaborate with experts on planning systems to compute action instances only by need.

4.2 Referring expressions

In our analysis of referring expressions, the tree t that introduces the new substitution nodes typically initializes the distractor sets with proper subsets of the entire domain. This allows us to generate succinct descriptions by encoding t's presuppositions as semantic requirements, and localizes the interactions between the referring expressions generated for different substitution nodes within t's action.

However, an important detail in the encoding of referring expressions above is that an individual a counts as a distractor for the role r if there is any tuple of values that satisfies the semantic requirement and has a in the r-component. This is correct, but can sometimes lead to overly complicated referring expressions. An example is the construction "X takes Y from Z", which presupposes that Y is in Z. In a scenario that involves multiple rabbits, multiple hats, and multiple individuals that are inside other individuals, but only one pair of a rabbit r inside a hat h, the expression "X takes the rabbit from the hat" is sufficient to refer uniquely to r and h (Stone and Webber, 1998). Our system would try to generate an expression for Y that suffices by itself to distinguish r from all distractors, and similarly for Z. We will explore this issue further in future work.

5 Conclusion

In this paper, we have shown how sentence generation with TAG grammars and semantic and pragmatic information can be encoded into PDDL. Our encoding is declarative in that it can be used with any correct planning algorithm, and explicit in that the actions capture the complete effect of a word on the syntactic, semantic, and local pragmatic goals. In terms of expressive power, it captures the core of SPUD, except for its inference capabilities.

This work is practically relevant because it opens up the possibility of using efficient planners to make generators faster and more flexible. Conversely, our PDDL problems are a challenge for current planners and open up NLG as an application domain that planning research itself can target.

Theoretically, our encoding provides a new framework for understanding and exploring the general relationships between language and action. It suggests new ways of going beyond SPUD's expressive power, to formulate utterances that describe and disambiguate concurrent real-world actions or exploit the dynamics of linguistic context within and across sentences.

Acknowledgments. This work was funded by a DFG research fellowship and the NSF grants HLC 0308121, IGERT 0549115, and HSD 0624191. We are indebted to Henry Kautz for his advice on planning systems, and to Owen Rambow, Bonnie Webber, and the anonymous reviewers for feedback.

References

D. Appelt. 1985. *Planning English Sentences*. Cambridge University Press, Cambridge England.

A. Blum and M. Furst. 1997. Fast planning through graph analysis. *Artificial Intelligence*, 90:281–300.

P. R. Cohen and C. R. Perrault. 1979. Elements of a plan-based theory of speech acts. *Cognitive Science*, 3(3):177–212.

R. Dale and E. Reiter. 1995. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 19.

D. DeVault, C. Rich, and C. Sidner. 2004. Natural language generation and discourse context: Computing distractor sets from the focus stack. In *Proc. FLAIRS*.

R. Fikes and N. Nilsson. 1971. STRIPS: A new approach in the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208.

P. Heeman and G. Hirst. 1995. Collaborating on referring expressions. *Computational Linguistics*, 21(3):351–382.

J. Hobbs, M. Stickel, D. Appelt, and P. Martin. 1993. Interpretation as abduction. *Artificial Intelligence*, 63:69–142.

J. Hoffmann and B. Nebel. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14.

A. Joshi and Y. Schabes. 1997. Tree-Adjoining Grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, chapter 2, pages 69–123. Springer-Verlag, Berlin.

H. Kautz and B. Selman. 1998. Blackbox: A new approach to the application of theorem proving to problem solving. In *Workshop Planning as Combinatorial Search, AIPS-98*.

A. Koller and K. Striegnitz. 2002. Generation as dependency parsing. In *Proc. 40th ACL*, Philadelphia.

D. V. McDermott. 2000. The 1998 AI Planning Systems Competition. *AI Magazine*, 21(2):35–55.

M. Stone and B. Webber. 1998. Textual economy through close coupling of syntax and semantics. In *Proc. INLG*.

M. Stone, C. Doran, B. Webber, T. Bleam, and M. Palmer. 2003. Microplanning with communicative intentions: The SPUD system. *Computational Intelligence*, 19(4):311–381.

R. Thomason and J. Hobbs. 1997. Interrelating interpretation and generation in an abductive framework. In *AAAI Fall Symposium on Communicative Action*.

GLEU: Automatic Evaluation of Sentence-Level Fluency

Andrew Mutton[*] **Mark Dras**[*] **Stephen Wan**[*,†] **Robert Dale**[*]

[*]Centre for Language Technology
Macquarie University
NSW 2109 Australia

[†]Information and Communication Technologies
CSIRO
NSW 2109 Australia

madras@ics.mq.edu.au

Abstract

In evaluating the output of language technology applications—MT, natural language generation, summarisation—automatic evaluation techniques generally conflate measurement of faithfulness to source content with fluency of the resulting text. In this paper we develop an automatic evaluation metric to estimate fluency alone, by examining the use of parser outputs as metrics, and show that they correlate with human judgements of generated text fluency. We then develop a machine learner based on these, and show that this performs better than the individual parser metrics, approaching a lower bound on human performance. We finally look at different language models for generating sentences, and show that while individual parser metrics can be 'fooled' depending on generation method, the machine learner provides a consistent estimator of fluency.

1 Introduction

Intrinsic evaluation of the output of many language technologies can be characterised as having at least two aspects: how well the generated text reflects the source data, whether it be text in another language for machine translation (MT), a natural language generation (NLG) input representation, a document to be summarised, and so on; and how well it conforms to normal human language usage. These two aspects are often made explicit in approaches to creating the text. For example, in statistical MT the translation model and the language model are treated separately, characterised as faithfulness and fluency respectively (as in the treatment in Jurafsky and Martin (2000)). Similarly, the ultrasummarisation model of Witbrock and Mittal (1999) consists of a content model, modelling the probability that a word in the source text will be in the summary, and a language model.

Evaluation methods can be said to fall into two categories: a comparison to gold reference, or an appeal to human judgements. Automatic evaluation methods carrying out a comparison to gold reference tend to conflate the two aspects of faithfulness and fluency in giving a goodness score for generated output. BLEU (Papineni et al., 2002) is a canonical example: in matching n-grams in a candidate translation text with those in a reference text, the metric measures faithfulness by counting the matches, and fluency by implicitly using the reference n-grams as a language model. Often we are interested in knowing the quality of the two aspects separately; many human judgement frameworks ask specifically for separate judgements on elements of the task that correspond to faithfulness and to fluency. In addition, the need for reference texts for an evaluation metric can be problematic, and intuitively seems unnecessary for characterising an aspect of text quality that is not related to its content source but to the use of language itself. It is a goal of this paper to provide an automatic evaluation method for fluency alone, without the use of a reference text.

One might consider using a metric based on language model probabilities for sentences: in eval-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 344–351,
Prague, Czech Republic, June 2007.

uating a language model on (already existing) test data, a higher probability for a sentence (and lower perplexity over a whole test corpus) indicates better language modelling; perhaps a higher probability might indicate a better sentence. However, here we are looking at generated sentences, which have been generated using their own language model, rather than human-authored sentences already existing in a test corpus; and so it is not obvious what language model would be an objective assessment of sentence naturalness. In the case of evaluating a single system, using the language model that generated the sentence will only confirm that the sentence does fit the language model; in situations such as comparing two systems which each generate text using a different language model, it is not obvious that there is a principled way of deciding on a fair language model. Quite a different idea was suggested in Wan et al. (2005), of using the grammatical judgement of a parser to assess fluency, giving a measure independent of the language model used to generate the text. The idea is that, assuming the parser has been trained on an appropriate corpus, the poor performance of the parser on one sentence relative to another might be an indicator of some degree of ungrammaticality and possibly disfluency. In that work, however, correlation with human judgements was left uninvestigated.

The goal of this paper is to take this idea and develop it. In Section 2 we look at some related work on metrics, in particular for NLG. In Section 3, we verify whether parser outputs can be used as estimators of generated sentence fluency by correlating them with human judgements. In Section 4, we propose an SVM-based metric using parser outputs as features, and compare its correlation against human judgements with that of the individual parsers. In Section 5, we investigate the effects on the various metrics from different types of language model for the generated text. Then in Section 6 we conclude.

2 Related Work

In terms of human evaluation, there is no uniform view on what constitutes the notion of fluency, or its relationship to grammaticality or similar concepts. We mention a few examples here to illustrate the range of usage. In MT, the 2005 NIST MT Evaluation Plan uses guidelines[1] for judges to assess 'adequacy' and 'fluency' on 5 point scales, where they are asked to provide intuitive reactions rather than pondering their decisions; for fluency, the scale descriptions are fairly vague (5: flawless English; 4: good English; 3: non-native English; 2: disfluent English; 1: incomprehensible) and instructions are short, with some examples provided in appendices. Zajic et al. (2002) use similar scales for summarisation. By contrast, Pan and Shaw (2004), for their NLG system SEGUE tied the notion of fluency more tightly to grammaticality, giving two human evaluators three grade options: good, minor grammatical error, major grammatical/pragmatic error. As a further contrast, the analysis of Coch (1996) was very comprehensive and fine-grained, in a comparison of three text-production techniques: he used 14 human judges, each judging 60 letters (20 per generation system), and required them to assess the letters for correct spelling, good grammar, rhythm and flow, appropriateness of tone, and several other specific characteristics of good text.

In terms of automatic evaluation, we are not aware of any technique that measures only fluency or similar characteristics, ignoring content, apart from that of Wan et al. (2005). Even in NLG, where, given the variability of the input representations (and hence difficulty in verifying faithfulness), it might be expected that such measures would be available, the available metrics still conflate content and form. For example, the metrics proposed in Bangalore et al. (2000), such as Simple Accuracy and Generation Accuracy, measure changes with respect to a reference string based on the idea of string-edit distance. Similarly, BLEU has been used in NLG, for example by Langkilde-Geary (2002).

3 Parsers as Evaluators

There are three parts to verifying the usefulness of parsers as evaluators: choosing the parsers and the metrics derived from them; generating some texts for human and parser evaluation; and, the key part, getting human judgements on these texts and correlating them with parser metrics.

[1] http://projects.ldc.upenn.edu/TIDES/Translation/TranAssessSpec.pdf

3.1 The Parsers

In testing the idea of using parsers to judge fluency, we use three parsers, from which we derive four parser metrics, to investigate the general applicability of the idea. Those chosen were the Connexor parser,[2] the Collins parser (Collins, 1999), and the Link Grammar parser (Grinberg et al., 1995). Each produces output that can be taken as representing degree of ungrammaticality, although this output is quite different for each.

Connexor is a commercially available dependency parser that returns head–dependant relations as well as stemming information, part of speech, and so on. In the case of an ungrammatical sentence, Connexor returns tree fragments, where these fragments are defined by transitive head–dependant relations: for example, for the sentence *Everybody likes big cakes do* it returns fragments for *Everybody likes big cakes* and for *do*. We expect that the number of fragments should correlate inversely with the quality of a sentence. For a metric, we normalise this number by the largest number of fragments for a given data set. (Normalisation matters most for the machine learner in Section 4.)

The Collins parser is a statistical chart parser that aims to maximise the probability of a parse using dynamic programming. The parse tree produced is annotated with log probabilities, including one for the whole tree. In the case of ungrammatical sentences, the parser will assign a low probability to any parse, including the most likely one. We expect that the log probability (becoming more negative as the sentence is less likely) should correlate positively with the quality of a sentence. For a metric, we normalise this by the most negative value for a given data set.

Like Connexor, the Link Grammar parser returns information about word relationships, forming links, with the proviso that links cannot cross and that in a grammatical sentence all links are indirectly connected. For an ungrammatical sentence, the parser will delete words until it can produce a parse; the number it deletes is called the 'null count'. We expect that this should correlate inversely with sentence quality. For a metric, we normalise this by the sentence length. In addition, the parser produces another variable possibly of interest. In generating a parse, the parser produces many candidates and rules some out by a posteriori constraints on valid parses. In its output the parser returns the number of invalid parses. For an ungrammatical sentence, this number may be higher; however, there may also be more parses. For a metric, we normalise this by the total number of parses found for the sentence. There is no strong intuition about the direction of correlation here, but we investigate it in any case.

[2] `http://www.connexor.com`

3.2 Text Generation Method

To test whether these parsers are able to discriminate sentence-length texts of varying degrees of fluency, we need first to generate texts that we expect will be discriminable in fluency quality ranging from good to very poor. Below we describe our method for generating text, and then our preliminary check on the discriminability of the data before giving them to human judges.

Our approach to generating 'sentences' of a fixed length is to take word sequences of different lengths from a corpus and glue them together probabilistically: the intuition is that a few longer sequences glued together will be more fluent than many shorter sequences. More precisely, to generate a sentence of length n, we take sequences of length l (such that l divides n), with sequence i of the form $w_{i,1} \ldots w_{i,l}$, where $w_{i,\_}$ is a word or punctuation mark. We start by selecting sequence 1, first by randomly choosing its first word according to the unigram probability $P(w_{1,1})$, and then the sequence uniformly randomly over all sequences of length l starting with $w_{1,1}$; we select subsequent sequences j ($2 \leq j \leq n/l$) randomly according to the bigram probability $P(w_{j,1} \,|\, w_{j-1,l})$. Taking as our corpus the Reuters corpus,[3] for length $n = 24$, we generate sentences for sequence sizes $l = 1, 2, 4, 8, 24$ as in Figure 1. So, for instance, the sequence-size 8 example was constructed by stringing together the three consecutive sequences of length 8 (*There . . . to*; *be . . . have*; *to . . .*.) taken from the corpus.

These examples, and others generated, appear to be of variable quality in accordance with our intuition. However, to confirm this prior to testing them

[3] `http://trec.nist.gov/data/reuters/reuters.html`

Extracted (Sequence-size 24)
Ginebra face Formula Shell in a sudden-death playoff on Sunday to decide who will face Alaska in a best-of-seven series for the title.
Sequence-size 8
There is some thinking in the government to be nearly as dramatic as some people have to be slaughtered to eradicate the epidemic.
Sequence-size 4
Most of Banharn's move comes after it can still be averted the crash if it should again become a police statement said.
Sequence-size 2
Massey said in line with losses, Nordbanken is well-placed to benefi t abuse was loaded with Czech prime minister Andris Shkele, said.
Sequence-size 1
The war we're here in a spokesman Jeff Sluman 86 percent jump that Spain to what was booked, express also said.

Figure 1: Sample sentences from the first trial

Description	Correlation
Small	0.10 to 0.29
Medium	0.30 to 0.49
Large	0.50 to 1.00

Table 1: Correlation coefficient interpretation

out for discriminability in a human trial, we wanted see whether they are discriminable by some method other than our own judgement. We used the parsers described in Section 3.1, in the hope of finding a non-zero correlation between the parser outputs and the sequence lengths.

Regarding the interpretation of the absolute value of (Pearson's) correlation coefficients, both here and in the rest of the paper, we adopt Cohen's scale (Cohen, 1988) for use in human judgements, given in Table 1; we use this as most of this work is to do with human judgements of fluency. For data, we generated 1000 sentences of length 24 for each sequence length $l = 1, 2, 3, 4, 6, 8, 24$, giving 7000 sentences in total. The correlations with the four parser outputs are as in Table 2, with the medium correlations for Collins and Link Grammar (nulled tokens) indicating that the sentences are indeed discriminable to some extent, and hence the approach is likely to be useful for generating sentences for human trials.

3.3 Human Judgements

The next step is then to obtain a set of human judgements for this data. Human judges can only be expected to judge a reasonably sized amount of data,

Metric	Corr.
Collins Parser	0.3101
Connexor	-0.2332
Link Grammar Nulled Tokens	-0.3204
Link Grammar Invalid Parses	0.1776
GLEU	0.4144

Table 2: Parser vs sequence size for original data set

so we first reduced the set of sequence sizes to be judged. To do this we determined for the 7000 generated sentences the scores according to the (arbitrarily chosen) Collins parser, and calculated the means for each sequence size and the 95% confidence intervals around these means. We then chose a subset of sequence sizes such that the confidence intervals did not overlap: $1, 2, 4, 8, 24$; the idea was that this would be likely to give maximally discriminable sentences. For each of these sequences sizes, we chose randomly 10 sentences from the initial set, giving a set for human judgement of size 50.

The judges consisted of twenty volunteers, all native English speakers without explicit linguistic training. We gave them general guidelines about what constituted fluency, mentioning that they should consider grammaticality but deliberately not giving detailed instructions on the manner for doing this, as we were interested in the level of agreement of intuitive understanding of fluency. We instructed them also that they should evaluate the sentence without considering its content, using *Colourless green ideas sleep furiously* as an example of a nonsensical but perfectly fluent sentence. The judges were then presented with the 50 sentences in random order, and asked to score the sentences according to their own scale, as in magnitude estimation (Bard et al., 1996); these scores were then normalised in the range [0,1].

Some judges noted that the task was difficult because of its subjectivity. Notwithstanding this subjectivity and variation in their approach to the task, the pairwise correlations between judges were high, as indicated by the maximum, minimum and mean values in Table 3, indicating that our assumption that humans had an intuitive notion of fluency and needed only minimal instruction was justified. Looking at mean scores for each sequence size, judges generally also ranked sentences by sequence size; see Figure 2. Comparing human judgement

Statistic	Corr.
Maximum correlation	0.8749
Minimum correlation	0.4710
Mean correlation	0.7040
Standard deviation	0.0813

Table 3: Data on correlation between humans

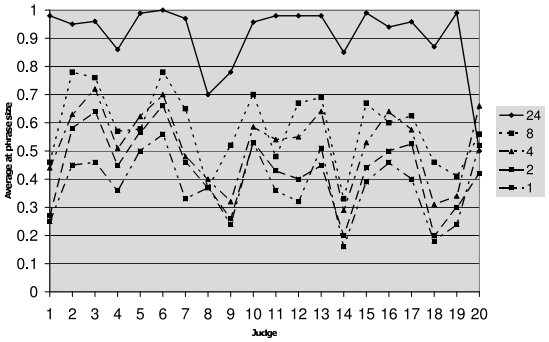


Figure 2: Mean scores for human judges

correlations against sequence size with the same correlations for the parser metrics (as for Table 2, but on the human trial data) gives Table 4, indicating that humans can also discriminate the different generated sentence types, in fact (not surprisingly) better than the automatic metrics.

Now, having both human judgement scores of some reliability for sentences, and scoring metrics from three parsers, we give correlations in Table 5. Given Cohen's interpretation, the Collins and Link Grammar (nulled tokens) metrics show moderate correlation, the Connexor metric almost so; the Link Grammar (invalid parses) metric correlation is by far the weakest. The consistency and magnitude of the first three parser metrics, however, lends support to the idea of Wan et al. (2005) to use something like these as indicators of generated sentence fluency. The aim of the next section is to build a better predictor than the individual parser metrics alone.

Metric	Corr.
Humans	0.6529
Collins Parser	0.4057
Connexor	-0.3804
Link Grammar Nulled Tokens	-0.3310
Link Grammar Invalid Parses	0.1619
GLEU	0.4606

Table 4: Correlation with sequence size for human trial data set

Metric	Corr.
Collins Parser	0.3057
Connexor	-0.3445
Link-Grammar Nulled Tokens	-0.2939
Link Grammar Invalid Parses	0.1854
GLEU	0.4014

Table 5: Correlation between metrics and human evaluators

4 An SVM-Based Metric

In MT, one problem with most metrics like BLEU is that they are intended to apply only to document-length texts, and any application to individual sentences is inaccurate and correlates poorly with human judgements. A neat solution to poor sentence-level evaluation proposed by Kulesza and Shieber (2004) is to use a Support Vector Machine, using features such as word error rate, to estimate sentence-level translation quality. The two main insights in applying SVMs here are, first, noting that human translations are generally good and machine translations poor, that binary training data can be created by taking the human translations as positive training instances and machine translations as negative ones; and second, that a non-binary metric of translation goodness can be derived by the distance from a test instance to the support vectors. In an empirical evaluation, Kulesza and Shieber found that their SVM gave a correlation of 0.37, which was an improvement of around half the gap between the BLEU correlations with the human judgements (0.25) and the lowest pairwise human inter-judge correlation (0.46) (Turian et al., 2003).

We take a similar approach here, using as features the four parser metrics described in Section 3. We trained an SVM,[4] taking as positive training data the 1000 instances of sentences of sequence length 24 (i.e. sentences extracted from the corpus) and as negative training data the 1000 sentences of sequence length 1. We call this learner GLEU.[5]

As a check on the ability of the GLEU SVM to distinguish these 'positive' sentences from 'negative' ones, we evaluated its classification accuracy on a (new) test set of size 300, split evenly between sentences of sequence length 24 and sequence length 1.

[4] We used the package SVM-light (Joachims, 1999).

[5] For GrammaticaLity Evaluation Utility.

This gave 81%, against a random baseline of 50%, indicating that the SVM can classify satisfactorily.

We now move from looking at classification accuracy to the main purpose of the SVM, using distance from support vector as a metric. Results are given for correlation of GLEU against sequence sizes for all data (Table 2) and for the human trial data set (Table 4); and also for correlation of GLEU against the human judges' scores (Table 5). This last indicates that GLEU correlates better with human judgements than any of the parsers individually, and is well within the 'moderate' range for correlation interpretation. In particular, for the GLEU–human correlation, the score of 0.4014 is approaching the minimum pairwise human correlation of 0.4710.

5 Different Text Generation Methods

The method used to generate text in Section 3.2 is a variation of the standard n-gram language model. A question that arises is: Are any of the metrics defined above strongly influenced by the type of language model used to generate the text? It may be the case, for example, that a parser implementation uses its own language model that predisposes it to favour a similar model in the text generation process. This is a phenomenon seen in MT, where BLEU seems to favour text that has been produced using a similar statistical n-gram language model over other symbolic models (Callison-Burch et al., 2006).

Our previous approach used only sequences of words concatenated together. To define some new methods for generating text, we introduced varying amounts of structure into the generation process.

5.1 Structural Generation Methods

PoStag In the first of these, we constructed a rough approximation of typical sentence grammar structure by taking bigrams over part-of-speech tags.[6] Then, given a string of PoS tags of length n, $t_1 \dots t_n$, we start by assigning the probabilities for the word in position 1, w_1, according to the conditional probability $P(w_1 \mid t_1)$. Then, for position j ($2 \leq j \leq n$), we assign to candidate words the value $P(w_j \mid t_j) \times P(w_j \mid w_{j-1})$ to score word sequences. So, for example, we might generate the PoS tag template **Det NN Adj Adv**, take all the words corresponding to each of these parts of speech, and combine bigram word sequence probability with the conditional probability of words with respect to these parts of speech. We then use a Viterbi-style algorithm to find the most likely word sequence.

In this model we violate the Markov assumption of independence in much the same way as Witbrock and Mittal (1999) in their combination of content and language model probabilities, by backtracking at every state in order to discourage repeated words and avoid loops.

Supertag This is a variant of the approach above, but using supertags (Bangalore and Joshi, 1999) instead of PoS tags. The idea is that the supertags might give a more fine-grained definition of structure, using partial trees rather than parts of speech.

CFG We extracted a CFG from the ∼10% of the Penn Treebank found in the NLTK-lite corpora.[7] This CFG was then augmented with productions derived from the PoS-tagged data used above. We then generated a template of length n pre-terminal categories using this CFG. To avoid loops we biased the selection towards terminals over non-terminals.

5.2 Human Judgements

We generated sentences according to a mix of the initial method of Section 3.2, for calibration, and the new methods above. We again used a sentence length of 24, and sequence lengths for the initial method of $l = 1, 8, 24$. A sample of sentences generated for each of these six types is in Figure 3.

For our data, we generated 1000 sentences per generation method, giving a corpus of 6000 sentences. For the human judgements we also again took 10 sentences per generation method, giving 60 sentences in total. The same judges were given the same instructions as previously.

Before correlating the human judges' scores and the parser outputs, it is interesting to look at how each parser treats the sentence generation methods, and how this compares with human ratings (Table 6). In particular, note that the Collins parser rates the PoStag- and Supertag-generated sentences more

[6] We used the supertagger of Bangalore and Joshi (1999).

[7] `http://nltk.sourceforge.net`

Extracted (Sequence-size 24)
After a near three-hour meeting and last-minute talks with President Lennart Meri, the Reform Party council voted overwhelmingly to leave the government.
Sequence-size 8
If Denmark is closely linked to the Euro Disney reported a net profi t of 85 million note: the fi gures were rounded off.
Sequence-size 1
Israelis there would seek approval for all-party peace now complain that this year, which shows demand following year and 56 billion pounds.
POS-tag, Viterbi-mapped
He said earlier the 9 years and holding company's government, including 69.62 points as a number of last year but market.
Supertag, Viterbi-mapped
That 97 saying he said in its shares of the market 74.53 percent, adding to allow foreign exchange: I think people.
Context-Free Grammar
The production moderated Chernomyrdin which leveled government back near own 52 over every a current at from the said by later the other.

Figure 3: Sample sentences from the second trial

sent. type	s-24	s-8	s-1	PoS	sup.	CFG
Collins	**0.52**	0.48	0.41	**0.60**	**0.57**	0.36
Connexor	0.12	0.16	0.24	0.26	0.25	0.43
LG (null)	0.02	0.06	0.10	0.09	0.11	0.18
LG (invalid)	0.78	0.67	0.56	0.62	0.66	0.53
GLEU	1.07	0.32	-0.96	0.28	-0.06	-2.48
Human	0.93	0.67	0.44	0.39	0.44	0.31

Table 6: Mean normalised scores per sentence type

highly even than real sentences (in bold). These are the two methods that use the Viterbi-style algorithm, suggesting that this probability maximisation has fooled the Collins parser. The pairwise correlation between judges was around the same on average as in Section 3.3, but with wider variation (Table 7).

The main results, determining the correlation of the various parser metrics plus GLEU against the new data, are in Table 8. This confirms the very variable performance of the Collins parser, which has dropped significantly. GLEU performs quite consistently here, this time a little behind the Link Grammar (nulled tokens) result, but still with a better correlation with human judgement than at least two

Statistic	Corr.
Maximum correlation	0.9048
Minimum correlation	0.3318
Mean correlation	0.7250
Standard deviation	0.0980

Table 7: Data on correlation between humans

Metric	Corr.
Collins Parser	0.1898
Connexor	-0.3632
Link-Grammar Nulled Tokens	-0.4803
Link Grammar Invalid Parses	0.1774
GLEU	0.4738

Table 8: Correlation between parsers and human evaluators on new human trial data

Metric	Corr.
Collins Parser	0.2313
Connexor	-0.2042
Link-Grammar Nulled Tokens	-0.1289
Link Grammar Invalid Parses	-0.0084
GLEU	0.4312

Table 9: Correlation between parsers and human evaluators on all human trial data

judges with each other. (Note also that the GLEU SVM was not retrained on the new sentence types.)

Looking at all the data together, however, is where GLEU particularly displays its consistency. Aggregating the old human trial data (Section 3.3) and the new data, and determining correlations against the metrics, we get the data in Table 9. Again the SVM's performance is consistent, but is now almost twice as high as its nearest alternative, Collins.

5.3 Discussion

In general, there is at least one parser that correlates quite well with the human judges for each sentence type. With well-structured sentences, the probabilistic Collins parser performs best; on sentences that are generated by a poor probabilistic model leading to poor structure, Link Grammar (nulled tokens) performs best. This supports the use of a machine learner taking as features outputs from several parser types; empirically this is confirmed by the large advantage GLEU has on overall data (Table 9).

The generated text itself from the Viterbi-based generators as implemented here is quite disappointing, given an expectation that introducing structure would make sentences more natural and hence lead to a range of sentence qualities. In hindsight, this is not so surprising; in generating the structure template, only sequences (over tags) of size 1 were used, which is perhaps why the human judges deemed them fairly close to sentences generated by the origi-

nal method using sequence size 1, the poorest of that initial data set.

6 Conclusion

In this paper we have investigated a new approach to evaluating the fluency of individual generated sentences. The notion of what constitutes fluency is an imprecise one, but trials with human judges have shown that even if it cannot be exactly defined, or even articulated by the judges, there is a high level of agreement about what is fluent and what is not. Given this data, metrics derived from parser outputs have been found useful for measuring fluency, correlating up to moderately well with these human judgements. A better approach is to combine these in a machine learner, as in our SVM GLEU, which outperforms individual parser metrics. Interestingly, we have found that the parser metrics can be fooled by the method of sentence generation; GLEU, however, gives a consistent estimate of fluency regardless of generation type; and, across all types of generated sentences examined in this paper, is superior to individual parser metrics by a large margin.

This all suggests that the approach has promise, but it needs to be developed further for pratical use. The SVM presented in this paper has only four features; more features, and in particular a wider range of parsers, should raise correlations. In terms of the data, we looked only at sentences generated with several parameters fixed, such as sentence length, due to our limited pool of judges. In future we would like to examine the space of sentence types more fully. In particular, we will look at predicting the fluency of near-human quality sentences. More generally, we would like to look also at how the approach of this paper would relate to a perplexity-based metric; how it compares against BLEU or similar measures as a predictor of fluency in a context where reference sentences are available; and whether GLEU might be useful in applications such as reranking of candidate sentences in MT.

Acknowledgements

We thank Ben Hutchinson and Mirella Lapata for discussions, and Srinivas Bangalore for the TAG supertagger. The second author acknowledges the support of ARC Discovery Grant DP0558852.

References

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

Srinivas Bangalore, Owen Rambow, and Steve Whittaker. 2000. Evaluation metrics for generation. In *Proceedings of the First International Natural Language Generation Conference (INLG2000)*, Mitzpe Ramon, Israel.

E. Bard, D. Robertson, and A. Sorace. 1996. Magnitude estimation and linguistic acceptability. *Language*, 72(1):32–68.

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.

José Coch. 1996. Evaluating and comparing three text-production strategies. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96)*, pages 249–254.

J. Cohen. 1988. *Statistical power analysis for the behavioral sciences*. Erlbaum, Hillsdale, NJ, US.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Dennis Grinberg, John Lafferty, and Daniel Sleator. 1995. A robus parsing algorithm for link grammars. In *Proceedings of the Fourth International Workshop on Parsing Technologies*.

Thorsten Joachims. 1999. *Making Large-Scale SVM Learning Practical*. MIT Press.

Daniel Jurafsky and James Martin. 2000. *Speech and Language Processing: An Introduction to Natural Languge Processing, Computational Linguistics, and Speech Recognition*. Prentice-Hall.

Alex Kulesza and Stuart 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*, Baltimore, MD, US.

Irene Langkilde-Geary. 2002. An empirical verifi cation of coverage and correctness for a general-purpose sentence generator. In *Proceedings of the International Natural Language Generation Conference (INLG) 2002*, pages 17–24.

Shimei Pan and James Shaw. 2004. Segue: A hybrid case-based surface natural language generator. In *Proceedings of the International Conference on Natural Language Generation (INLG) 2004*, pages 130–140.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. Technical Report RC22176, IBM.

Joseph Turian, Luke Shen, and I. Dan Melamed. 2003. Evaluation of Machine Translation and its evaluation. In *Proceedings of MT Summit IX*, pages 23–28.

Stephen Wan, Robert Dale, Mark Dras, and Cécile Paris. 2005. Searching for grammaticality: Propagating dependencies in the Viterbi algorithm. In *Proceedings of the 10th European Natural Language Processing Wworkshop*, Aberdeen, UK.

Michael Witbrock and Vibhu Mittal. 1999. Ultra-summarization: A statistical approach to generating highly condensed non-executive summaries. In *Proceedings of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR'99)*.

David Zajic, Bonnie Dorr, and Richard Schwartz. 2002. Automatic headline generation for newspaper stories. In *Proceedings of the ACL-2002 Workshop on Text Summarization (DUC2002)*, pages 78–85.

Conditional Modality Fusion for Coreference Resolution

Jacob Eisenstein and Randall Davis
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139 USA
{jacobe,davis}@csail.mit.edu

Abstract

Non-verbal modalities such as gesture can improve processing of spontaneous spoken language. For example, similar hand gestures tend to predict semantic similarity, so features that quantify gestural similarity can improve semantic tasks such as coreference resolution. However, not all hand movements are informative gestures; psychological research has shown that speakers are more likely to gesture meaningfully when their speech is ambiguous. Ideally, one would attend to gesture only in such circumstances, and ignore other hand movements. We present *conditional modality fusion*, which formalizes this intuition by treating the informativeness of gesture as a hidden variable to be learned jointly with the class label. Applied to coreference resolution, conditional modality fusion significantly outperforms both early and late modality fusion, which are current techniques for modality combination.

1 Introduction

Non-verbal modalities such as gesture and prosody can increase the robustness of NLP systems to the inevitable disfluency of spontaneous speech. For example, consider the following excerpt from a dialogue in which the speaker describes a mechanical device:

> "So this moves up, and it – everything moves up. And this top one clears this area here, and goes all the way up to the top."

The references in this passage are difficult to disambiguate, but the gestures shown in Figure 1 make the meaning more clear. However, non-verbal modalities are often noisy, and their interactions with speech are complex (McNeill, 1992). Gesture, for example, is sometimes communicative, but other times merely distracting. While people have little difficulty distinguishing between meaningful gestures and irrelevant hand motions (e.g., self-touching, adjusting glasses) (Goodwin and Goodwin, 1986), NLP systems may be confused by such seemingly random movements. Our goal is to include non-verbal features only in the specific cases when they are helpful and necessary.

We present a model that learns in an unsupervised fashion when non-verbal features are useful, allowing it to gate the contribution of those features. The relevance of the non-verbal features is treated as a hidden variable, which is learned jointly with the class label in a conditional model. We demonstrate that this improves performance on binary coreference resolution, the task of determining whether a noun phrases refers to a single semantic entity. Conditional modality fusion yields a relative increase of 73% in the contribution of hand-gesture features. The model is not specifically tailored to gesture-speech integration, and may also be applicable to other non-verbal modalities.

2 Related work

Most of the existing work on integrating non-verbal features relates to prosody. For example, Shriberg et al. (2000) explore the use of prosodic features for sentence and topic segmentation. The first modal-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 352–359,
Prague, Czech Republic, June 2007.

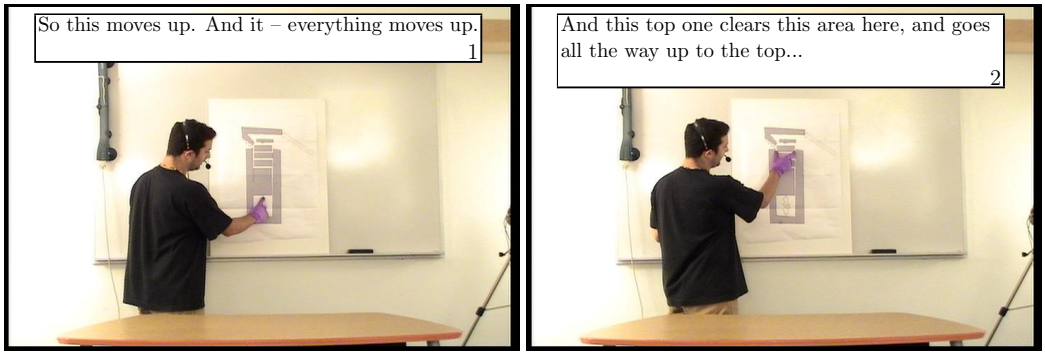


Figure 1: An example where gesture helps to disambiguate meaning.

ity combination technique that they consider trains a single classifier with all modalities combined into a single feature vector; this is sometimes called "early fusion." Shriberg et al. also consider training separate classifiers and combining their posteriors, either through weighted addition or multiplication; this is sometimes called "late fusion." Late fusion is also employed for gesture-speech combination in (Chen et al., 2004). Experiments in both (Shriberg et al., 2000) and (Kim et al., 2004) find no conclusive winner among early fusion, additive late fusion, and multiplicative late fusion.

Toyama and Horvitz (2000) introduce a Bayesian network approach to modality combination for speaker identification. As in late fusion, modality-specific classifiers are trained independently. However, the Bayesian approach also learns to predict the reliability of each modality on a given instance, and incorporates this information into the Bayes net. While more flexible than the interpolation techniques described in (Shriberg et al., 2000), training modality-specific classifiers separately is still suboptimal compared to training them jointly, because independent training of the modality-specific classifiers forces them to account for data that they cannot possibly explain. For example, if the speaker is not gesturing meaningfully, it is counterproductive to train a gesture-modality classifier on the features at this instant; doing so can lead to overfitting and poor generalization.

Our approach combines aspects of both early and late fusion. As in early fusion, classifiers for all modalities are trained jointly. But as in Toyama and Horvitz's Bayesian late fusion model, modalities can be weighted based on their predictive power for specific instances. In addition, our model is trained to maximize conditional likelihood, rather than joint likelihood.

3 Conditional modality fusion

The goal of our approach is to learn to weight the non-verbal features $\mathbf{x}_{nv}$ only when they are relevant. To do this, we introduce a hidden variable $m \in \{-1, 1\}$, which governs whether the non-verbal features are included. $p(m)$ is conditioned on a subset of features $\mathbf{x}_m$, which may belong to any modality; $p(m|\mathbf{x}_m)$ is learned jointly with the class label $p(y|\mathbf{x})$, with $y \in \{-1, 1\}$. For our coreference resolution model, y corresponds to whether a given pair of noun phrases refers to the same entity.

In a log-linear model, parameterized by weights $\mathbf{w}$, we have:

$$\begin{aligned} p(y|\mathbf{x};\mathbf{w}) &= \sum_m p(y, m|\mathbf{x};\mathbf{w}) \\ &= \frac{\sum_m exp(\psi(y, m, \mathbf{x};\mathbf{w}))}{\sum_{y',m} exp(\psi(y', m, \mathbf{x};\mathbf{w}))}. \end{aligned}$$

Here, ψ is a potential function representing the compatibility between the label y, the hidden variable m, and the observations $\mathbf{x}$; this potential is parameterized by a vector of weights, $\mathbf{w}$. The numerator expresses the compatibility of the label y and observations $\mathbf{x}$, summed over all possible values of the hidden variable m. The denominator sums over both m and all possible labels y', yielding the conditional probability $p(y|\mathbf{x};\mathbf{w})$. The use of hidden variables

in a conditionally-trained model follows (Quattoni et al., 2004).

This model can be trained by a gradient-based optimization to maximize the conditional log-likelihood of the observations. The unregularized log-likelihood and gradient are given by:

$$l(\mathbf{w}) = \sum_i \ln(p(y_i|\mathbf{x}_i; w)) \quad (1)$$

$$= \sum_i \ln \frac{\sum_m exp(\psi(y_i, m, \mathbf{x}_i; \mathbf{w}))}{\sum_{y',m} exp(\psi(y', m, \mathbf{x}_i; \mathbf{w}))} \quad (2)$$

$$\frac{\partial l_i}{\partial w_j} = \sum_m p(m|y_i, \mathbf{x}_i; \mathbf{w}) \frac{\partial}{\partial w_j} \psi(y_i, m, \mathbf{x}_i; \mathbf{w}) - \sum_{y',m} p(m, y'|\mathbf{x}_i; \mathbf{w}) \frac{\partial}{\partial w_j} \psi(y', m, \mathbf{x}_i; \mathbf{w})$$

The form of the potential function ψ is where our intuitions about the role of the hidden variable are formalized. Our goal is to include the non-verbal features $\mathbf{x}_{nv}$ only when they are relevant; consequently, the weight for these features should go to zero for some settings of the hidden variable m. In addition, *verbal* language is different when used in combination with meaningful non-verbal communication than when it is used unimodally (Kehler, 2000; Melinger and Levelt, 2004). Thus, we learn a different set of feature weights for each case: $\mathbf{w}_{v,1}$ when the non-verbal features are included, and $\mathbf{w}_{v,2}$ otherwise. The formal definition of the potential function for conditional modality fusion is:

$$\psi(y, m, \mathbf{x}; \mathbf{w}) \equiv \begin{cases} y(\mathbf{w}_{v,1}^T \mathbf{x}_v + \mathbf{w}_{nv}^T \mathbf{x}_{nv}) + \mathbf{w}_m^T \mathbf{x}_m & m = 1 \\ y\mathbf{w}_{v,2}^T \mathbf{x}_v - \mathbf{w}_m^T \mathbf{x}_m & m = -1. \end{cases} \quad (3)$$

4 Application to coreference resolution

We apply conditional modality fusion to coreference resolution – the problem of partitioning the noun phrases in a document into clusters, where all members of a cluster refer to the same semantic entity. Coreference resolution on text datasets is well-studied (e.g., (Cardie and Wagstaff, 1999)). This prior work provides the departure point for our investigation of coreference resolution on spontaneous and unconstrained speech and gesture.

4.1 Form of the model

The form of the model used in this application is slightly different from that shown in Equation 3. When determining whether two noun phrases corefer, the features at each utterance must be considered. For example, if we are to compare the similarity of the gestures that accompany the two noun phrases, it should be the case that gesture is relevant during *both* time periods.

For this reason, we create two hidden variables, m_1 and m_2; they indicate the relevance of gesture over the first (antecedent) and second (anaphor) noun phrases, respectively. Since gesture similarity is only meaningful if the gesture is relevant during *both* NPs, the gesture features are included only if $m_1 = m_2 = 1$. Similarly, the linguistic feature weights $\mathbf{w}_{v,1}$ are used when $m_1 = m_2 = 1$; otherwise the weights $\mathbf{w}_{v,2}$ are used. This yields the model shown in Equation 4.

The vector of meta features $\mathbf{x}_{m_1}$ includes all single-phrase verbal and gesture features from Table 1, computed at the antecedent noun phrase; $\mathbf{x}_{m_2}$ includes the single-phrase verbal and gesture features, computed at the anaphoric noun phrase. The label-dependent verbal features $\mathbf{x}_v$ include both pairwise and single phrase verbal features from the table, while the label-dependent non-verbal features $\mathbf{x}_{nv}$ include only the pairwise gesture features. The single-phrase non-verbal features were not included because they were not thought to be informative as to whether the associated noun-phrase would participate in coreference relations.

4.2 Verbal features

We employ a set of verbal features that is similar to the features used by state-of-the-art coreference resolution systems that operate on text (e.g., (Cardie and Wagstaff, 1999)). Pairwise verbal features include: several string-match variants; distance features, measured in terms of the number of intervening noun phrases and sentences between the candidate NPs; and some syntactic features that can be computed from part of speech tags. Single-phrase verbal features describe the type of the noun phrase (definite, indefinite, demonstrative (e.g., *this ball*), or pronoun), the number of times it appeared in the document, and whether there were any adjecti-

$$\psi(y, m_1, m_2, \mathbf{x}; \mathbf{w}) \equiv \begin{cases} y(\mathbf{w}_{v,1}^T \mathbf{x}_v + \mathbf{w}_{nv}^T \mathbf{x}_{nv}) + m_1 \mathbf{w}_m^T \mathbf{x}_{m_1} + m_2 \mathbf{w}_m^T \mathbf{x}_{m_2}, & m_1 = m_2 = 1 \\ y\mathbf{w}_{v,2}^T \mathbf{x}_v + m_1 \mathbf{w}_m^T \mathbf{x}_{m_1} + m_2 \mathbf{w}_m^T \mathbf{x}_{m_2}, & \text{otherwise.} \end{cases} \quad (4)$$

val modifiers. The continuous-valued features were binned using a supervised technique (Fayyad and Irani, 1993).

Note that some features commonly used for coreference on the MUC and ACE corpora are not applicable here. For example, gazetteers listing names of nations or corporations are not relevant to our corpus, which focuses on discussions of mechanical devices (see section 5). Because we are working from transcripts rather than text, features dependent on punctuation and capitalization, such as apposition, are also not applicable.

4.3 Non-verbal features

Our non-verbal features attempt to capture similarity between the speaker's hand gestures; similar gestures are thought to suggest semantic similarity (McNeill, 1992). For example, two noun phrases may be more likely to corefer if they are accompanied by identically-located pointing gestures. In this section, we describe features that quantify various aspects of gestural similarity.

The most straightforward measure of similarity is the Euclidean distance between the average hand position during each noun phrase – we call this the FOCUS-DISTANCE feature. Euclidean distance captures cases in which the speaker is performing a gestural "hold" in roughly the same location (McNeill, 1992).

However, Euclidean distance may not correlate directly with semantic similarity. For example, when gesturing at a detailed part of a diagram, very small changes in hand position may be semantically meaningful, while in other regions positional similarity may be defined more loosely. Ideally, we would compute a semantic feature capturing the *object* of the speaker's reference (e.g., "the red block"), but this is not possible in general, since a complete taxonomy of all possible objects of reference is usually unknown. Instead, we use a hidden Markov model (HMM) to perform a spatio-temporal clustering on hand position and velocity. The SAME-CLUSTER feature reports whether the hand positions during two noun phrases were usually grouped in the same cluster by the HMM. JS-DIV reports the Jensen-Shannon divergence, a continuous-valued feature used to measure the similarity in cluster assignment probabilities between the two gestures (Lin, 1991).

The gesture features described thus far capture the similarity between static gestures; that is, gestures in which the hand position is nearly constant. However, these features do not capture the similarity between gesture trajectories, which may also be used to communicate meaning. For example, a description of two identical motions might be expressed by very similar gesture trajectories. To measure the similarity between gesture trajectories, we use dynamic time warping (Huang et al., 2001), which gives a similarity metric for temporal data that is invariant to speed. This is reported in the DTW-DISTANCE feature.

All features are computed from hand and body pixel coordinates, which are obtained via computer vision; our vision system is similar to (Deutscher et al., 2000). The feature set currently supports only single-hand gestures, using the hand that is farthest from the body center. As with the verbal feature set, supervised binning was applied to the continuous-valued features.

4.4 Meta features

The role of the meta features is to determine whether the gesture features are relevant at a given point in time. To make this determination, both verbal and non-verbal features are applied; the only requirement is that they be computable at a single instant in time (unlike features that measure the similarity between two NPs or gestures).

Verbal meta features Meaningful gesture has been shown to be more frequent when the associated speech is ambiguous (Melinger and Levelt, 2004). Kehler finds that fully-specified noun phrases are less likely to receive multimodal support (Kehler, 2000). These findings lead us to expect that pro-

Pairwise verbal features	
edit-distance	a numerical measure of the string similarity between the two NPs
exact-match	true if the two NPs have identical surface forms
str-match	true if the NPs are identical after removing articles
nonpro-str	true if i and j are not pronouns, and str-match is true
pro-str	true if i and j are pronouns, and str-match is true
j-substring-i	true if the anaphor j is a substring of the antecedent i
i-substring-j	true if i is a substring of j
overlap	true if there are any shared words between i and j
np-dist	the number of noun phrases between i and j in the document
sent-dist	the number of sentences between i and j in the document
both-subj	true if both i and j precede the first verb of their sentences
same-verb	true if the first verb in the sentences for i and j is identical
number-match	true if i and j have the same number
Single-phrase verbal features	
pronoun	true if the NP is a pronoun
count	number of times the NP appears in the document
has-modifiers	true if the NP has adjective modifiers
indef-np	true if the NP is an indefinite NP (e.g., *a fish*)
def-np	true if the NP is a definite NP (e.g., *the scooter*)
dem-np	true if the NP begins with *this*, *that*, *these*, or *those*
lexical features	lexical features are defined for the most common pronouns: *it*, *that*, *this*, and *they*
Pairwise gesture features	
focus-distance	the Euclidean distance in pixels between the average hand position during the two NPs
DTW-agreement	a measure of the agreement of the hand-trajectories during the two NPs, computed using dynamic time warping
same-cluster	true if the hand positions during the two NPs fall in the same cluster
JS-div	the Jensen-Shannon divergence between the cluster assignment likelihoods
Single-phrase gesture features	
dist-to-rest	distance of the hand from rest position
jitter	sum of instantaneous motion across NP
speed	total displacement over NP, divided by duration
rest-cluster	true if the hand is usually in the cluster associated with rest position
movement-cluster	true if the hand is usually in the cluster associated with movement

Table 1: The feature set

nouns should be likely to co-occur with meaningful gestures, while definite NPs and noun phrases that include adjectival modifiers should be unlikely to do so. To capture these intuitions, all single-phrase verbal features are included as meta features.

Non-verbal meta features Research on gesture has shown that semantically meaningful hand motions usually take place away from "rest position," which is located at the speaker's lap or sides (McNeill, 1992). Effortful movements away from these default positions can thus be expected to predict that gesture is being used to communicate. We identify rest position as the center of the body on the x-axis, and at a fixed, predefined location on the y-axis. The DIST-TO-REST feature computes the average Euclidean distance of the hands from the rest position, over the duration of the NP.

As noted in the previous section, a spatio-temporal clustering was performed on the hand positions and velocities, using an HMM. The REST-CLUSTER feature takes the value "true" iff the most frequently occupied cluster during the NP is the cluster closest to rest position. In addition, parameter tying in the HMM forces all clusters but one to represent static hold, with the remaining cluster accounting for the transition movements between holds. Only this last cluster is permitted to have an expected non-zero speed; if the hand is most frequently in this cluster during the NP, then the MOVEMENT-CLUSTER feature takes the value "true."

4.5 Implementation

The objective function (Equation 1) is optimized using a Java implementation of L-BFGS, a quasi-Newton numerical optimization technique (Liu and Nocedal, 1989). Standard L2-norm regularization is employed to prevent overfitting, with cross-validation to select the regularization constant. Although standard logistic regression optimizes a convex objective, the inclusion of the hidden variable renders our objective non-convex. Thus, convergence to a global minimum is not guaranteed.

5 Evaluation setup

Dataset Our dataset consists of sixteen short dialogues, in which participants explained the behavior

of mechanical devices to a friend. There are nine different pairs of participants; each contributed two dialogues, with two thrown out due to recording errors. One participant, the "speaker," saw a short video describing the function of the device prior to the dialogue; the other participant was tested on comprehension of the device's behavior after the dialogue. The speaker was given a pre-printed diagram to aid in the discussion. For simplicity, only the speaker's utterances were included in these experiments.

The dialogues were limited to three minutes in duration, and most of the participants used the entire allotted time. "Markable" noun phrases – those that are permitted to participate in coreference relations – were annotated by the first author, in accordance with the MUC task definition (Hirschman and Chinchor, 1997). A total of 1141 "markable" NPs were transcribed, roughly half the size of the MUC6 development set, which includes 2072 markable NPs over 30 documents.

Evaluation metric Coreference resolution is often performed in two phases: a binary classification phase, in which the likelihood of coreference for each pair of noun phrases is assessed; and a partitioning phase, in which the clusters of mutually-coreferring NPs are formed, maximizing some global criterion (Cardie and Wagstaff, 1999). Our model does not address the formation of noun-phrase clusters, but only the question of whether each pair of noun phrases in the document corefer. Consequently, we evaluate only the binary classification phase, and report results in terms of the area under the ROC curve (AUC). As the small size of the corpus did not permit dedicated test and development sets, results are computed using leave-one-out cross-validation, with one fold for each of the sixteen documents in the corpus.

Baselines Three types of baselines were compared to our conditional modality fusion (CMF) technique:

- **Early fusion**. The early fusion baseline includes all features in a single vector, ignoring modality. This is equivalent to standard maximum-entropy classification. Early fusion is implemented with a conditionally-trained linear classifier; it uses the same code as the CMF model, but always includes all features.
- **Late fusion**. The late fusion baselines train separate classifiers for gesture and speech, and then combine their posteriors. The modality-specific classifiers are conditionally-trained linear models, and again use the same code as the CMF model. For simplicity, a parameter sweep identifies the interpolation weights that maximize performance on the test set. Thus, it is likely that these results somewhat overestimate the performance of these baseline models. We report results for both additive and multiplicative combination of posteriors.
- **No fusion**. These baselines include the features from only a single modality, and again build a conditionally-trained linear classifier. Implementation uses the same code as the CMF model, but weights on features outside the target modality are forced to zero.

Although a comparison with existing state-of-the-art coreference systems would be ideal, all such available systems use verbal features that are inapplicable to our dataset, such as punctuation, capitalization, and gazetteers. The verbal features that we have included are a representative sample from the literature (e.g., (Cardie and Wagstaff, 1999)). The "no fusion, verbal features only" baseline thus provides a reasonable representation of prior work on coreference, by applying a maximum-entropy classifier to this set of typical verbal features.

Parameter tuning Continuous features are binned separately for each cross-validation fold, using only the training data. The regularization constant is selected by cross-validation within each training subset.

6 Results

Conditional modality fusion outperforms all other approaches by a statistically significant margin (Table 2). Compared with early fusion, CMF offers an absolute improvement of 1.20% in area under the ROC curve (AUC).[1] A paired t-test shows that this

[1] AUC quantifies the ranking accuracy of a classifier. If the AUC is 1, all positively-labeled examples are ranked higher than all negative-labeled ones.

model	AUC
Conditional modality fusion	**.8226**
Early fusion	.8109
Late fusion, multiplicative	.8103
Late fusion, additive	.8068
No fusion (verbal features only)	.7945
No fusion (gesture features only)	.6732

Table 2: Results, in terms of areas under the ROC curve

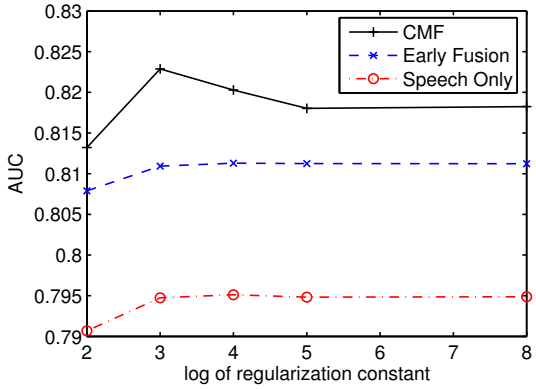


Figure 2: Conditional modality fusion is robust to variations in the regularization constant.

result is statistically significant ($p < .002, t(15) = 3.73$). CMF obtains higher performance on fourteen of the sixteen test folds. Both additive and multiplicative late fusion perform on par with early fusion.

Early fusion with gesture features is superior to unimodal verbal classification by an absolute improvement of 1.64% AUC ($p < 4 * 10^{-4}, t(15) = 4.45$). Thus, while gesture features improve coreference resolution on this dataset, their effectiveness is increased by a relative 73% when conditional modality fusion is applied. Figure 2 shows how performance varies with the regularization constant.

7 Discussion

The feature weights learned by the system to determine coreference largely confirm our linguistic intuitions. Among the textual features, a large positive weight was assigned to the string match features, while a large negative weight was assigned to features such as number incompatibility (i.e., singular versus plural). The system also learned that gestures with similar hand positions and trajectories were likely to indicate coreferring noun phrases; all of our similarity metrics were correlated positively with coreference. A chi-squared analysis found that the EDIT DISTANCE was the most informative verbal feature. The most informative gesture feature was DTW-AGREEMENT feature, which measures the similarity between gesture trajectories.

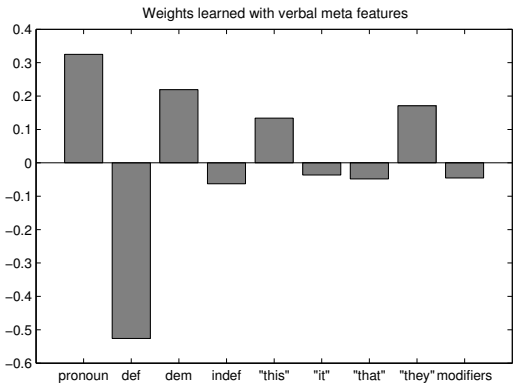


Figure 3: Weights for verbal meta features

As described in section 4, both textual and gestural features are used to determine whether the gesture is relevant. Among textual features, definite and indefinite noun phrases were assigned negative weights, suggesting gesture would not be useful to disambiguate coreference for such NPs. Pronouns were assigned positive weights, with "this" and the much less frequently used "they" receiving the strongest weights. "It" and "that" received lower weights; we observed that these pronouns were frequently used to refer to the immediately preceding noun phrase, so multimodal support was often unnecessary. Last, we note that NPs with adjectival modifiers were assigned negative weights, supporting the finding of (Kehler, 2000) that fully-specified NPs are less likely to receive multimodal support. A summary of the weights assigned to the verbal meta features is shown in Figure 3. Among gesture meta features, the weights learned by the system indicate that non-moving hand gestures away from the body are most likely to be informative in this dataset.

8 Future work

We have assumed that the relevance of gesture to semantics is dependent only on the currently available features, and not conditioned on prior history. In reality, meaningful gestures occur over contiguous blocks of time, rather than at randomly distributed instances. Indeed, the psychology literature describes a finite-state model of gesture, proceeding from "preparation," to "stroke," "hold," and then "retraction" (McNeill, 1992). These units are called *movement phases*. The relevance of various gesture features may be expected to depend on the movement phase. During strokes, the trajectory of the gesture may be the most relevant feature, while during holds, static features such as hand position and hand shape may dominate; during preparation and retraction, gesture features are likely to be irrelevant.

The identification of these movement phases should be independent of the specific problem of coreference resolution. Thus, additional labels for other linguistic phenomena (e.g., topic segmentation, disfluency) could be combined into the model. Ideally, each additional set of labels would transfer performance gains to the other labeling problems.

9 Conclusions

We have presented a new method for combining multiple modalities, which we feel is especially relevant to non-verbal modalities that are used to communicate only intermittently. Our model treats the relevance of the non-verbal modality as a hidden variable, learned jointly with the class labels. Applied to coreference resolution, this model yields a relative increase of 73% in the contribution of the gesture features. This gain is attained by identifying instances in which gesture features are especially relevant, and weighing their contribution more heavily. We next plan to investigate models with a temporal component, so that the behavior of the hidden variable is governed by a finite-state transducer.

Acknowledgments We thank Aaron Adler, Regina Barzilay, S. R. K. Branavan, Sonya Cates, Erdong Chen, Michael Collins, Lisa Guttentag, Michael Oltmans, and Tom Ouyang. This research is supported in part by MIT Project Oxygen.

References

Claire Cardie and Kiri Wagstaff. 1999. Noun phrase coreference as clustering. In *Proceedings of EMNLP*, pages 82–89.

Lei Chen, Yang Liu, Mary P. Harper, and Elizabeth Shriberg. 2004. Multimodal model integration for sentence unit detection. In *Proceedings of ICMI*, pages 121–128.

Jonathan Deutscher, Andrew Blake, and Ian Reid. 2000. Articulated body motion capture by annealed particle filtering. In *Proceedings of CVPR*, volume 2, pages 126–133.

Usama M. Fayyad and Keki B. Irani. 1993. Multi-interval discretization of continuousvalued attributes for classification learning. In *Proceedings of IJCAI-93*, volume 2, pages 1022–1027. Morgan Kaufmann.

M.H. Goodwin and C. Goodwin. 1986. Gesture and coparticipation in the activity of searching for a word. *Semiotica*, 62:51–75.

Lynette Hirschman and Nancy Chinchor. 1997. MUC-7 coreference task definition. In *Proceedings of the Message Understanding Conference*.

Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. 2001. *Spoken Language Processing*. Prentice Hall.

Andrew Kehler. 2000. Cognitive status and form of reference in multimodal human-computer interaction. In *Proceedings of AAAI*, pages 685–690.

Joungbum Kim, Sarah E. Schwarm, and Mari Osterdorf. 2004. Detecting structural metadata with decision trees and transformation-based learning. In *Proceedings of HLT-NAACL'04*. ACL Press.

Jianhua Lin. 1991. Divergence measures based on the shannon entropy. *IEEE transactions on information theory*, 37:145–151.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528.

David McNeill. 1992. *Hand and Mind*. The University of Chicago Press.

Alissa Melinger and Willem J. M. Levelt. 2004. Gesture and communicative intention of the speaker. *Gesture*, 4(2):119–141.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. 2004. Conditional random fields for object recognition. In *Neural Information Processing Systems*, pages 1097–1104.

Elizabeth Shriberg, Andreas Stolcke, Dilek Hakkani-Tur, and Gokhan Tur. 2000. Prosody-based automatic segmentation of speech into sentences and topics. *Speech Communication*, 32.

Kentaro Toyama and Eric Horvitz. 2000. Bayesian modality fusion: Probabilistic integration of multiple vision algorithms for head tracking. In *Proceedings of ACCV '00, Fourth Asian Conference on Computer Vision*.

The Utility of a Graphical Representation of Discourse Structure in Spoken Dialogue Systems

Mihai Rotaru
University of Pittsburgh
Pittsburgh, USA
mrotaru@cs.pitt.edu

Diane J. Litman
University of Pittsburgh
Pittsburgh, USA
litman@cs.pitt.edu

Abstract

In this paper we explore the utility of the Navigation Map (**NM**), a graphical representation of the discourse structure. We run a user study to investigate if users perceive the NM as helpful in a tutoring spoken dialogue system. From the users' perspective, our results show that the NM presence allows them to better identify and follow the tutoring plan and to better integrate the instruction. It was also easier for users to concentrate and to learn from the system if the NM was present. Our preliminary analysis on objective metrics further strengthens these findings.

1 Introduction

With recent advances in spoken dialogue system technologies, researchers have turned their attention to more complex domains (e.g. tutoring (Litman and Silliman, 2004; Pon-Barry et al., 2006), technical support (Acomb et al., 2007), medication assistance (Allen et al., 2006)). These domains bring forward new challenges and issues that can affect the usability of such systems: increased task complexity, user's lack of or limited task knowledge, and longer system turns.

In typical information access dialogue systems, the task is relatively simple: get the information from the user and return the query results with minimal complexity added by confirmation dialogues. Moreover, in most cases, users have knowledge about the task. However, in complex domains things are different. Take for example tutoring. A tutoring dialogue system has to discuss concepts, laws and relationships and to engage in complex subdialogues to correct user misconceptions. In addition, it is very likely that users of such systems are not familiar or are only partially familiar with the tutoring topic. The length of system turns can also be affected as these systems need to make explicit the connections between parts of the underlying task.

Thus, interacting with such systems can be characterized by an increased user cognitive load associated with listening to often lengthy system turns and the need to integrate the current information to the discussion overall (Oviatt et al., 2004).

We hypothesize that one way to reduce the user's cognitive load is to make explicit two pieces of information: the purpose of the current system turn, and how the system turn relates to the overall discussion. This information is implicitly encoded in the intentional structure of a discourse as proposed in the Grosz & Sidner theory of discourse (Grosz and Sidner, 1986).

Consequently, in this paper we propose using a graphical representation of the discourse structure as a way of improving the performance of complex-domain dialogue systems (note that graphical output is required). We call it the **Navigation Map** (**NM**). The NM is a dynamic representation of the discourse segment hierarchy and the discourse segment purpose information enriched with several features (Section 3). To make a parallel with geography, as the system "navigates" with the user through the domain, the NM offers a cartographic view of the discussion. While a somewhat similar graphical representation of the discourse structure has been explored in one previous study (Rich and Sidner, 1998), to our knowledge we are the first to test its benefits (see Section 6).

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 360–367,
Prague, Czech Republic, June 2007.

As a first step towards understanding the NM effects, here we focus on investigating whether users prefer a system with the NM over a system without the NM and, if yes, what are the NM usage patterns. We test this in a speech based computer tutor (Section 2). We run a within-subjects user study in which users interacted with the system both with and without the NM (Section 4).

Our analysis of the users' subjective evaluation of the system indicates that users prefer the version of the system with the NM over the version without the NM on several dimensions. The NM presence allows the users to better identify and follow the tutoring plan and to better integrate the instruction. It was also easier for users to concentrate and to learn from the system if the NM was present. Our preliminary analysis on objective metrics further strengthens these findings.

2 ITSPOKE

ITSPOKE (Litman and Silliman, 2004) is a state-of-the-art tutoring spoken dialogue system for conceptual physics. When interacting with ITSPOKE, users first type an essay answering a qualitative physics problem using a graphical user interface. ITSPOKE then engages the user in spoken dialogue (using head-mounted microphone input and speech output) to correct misconceptions and elicit more complete explanations, after which the user revises the essay, thereby ending the tutoring or causing another round of tutoring/essay revision.

All dialogues with ITSPOKE follow a question-answer format (i.e. system initiative): ITSPOKE asks a question, users answer and then the process is repeated. Deciding what question to ask, in what order and when to stop is hand-authored beforehand in a hierarchical structure. Internally, system questions are grouped in *question segments*.

In Figure 1, we show the transcript of a sample interaction with ITSPOKE. The system is discussing the problem listed in the upper right corner of the figure and it is currently asking the question $Tutor_5$. The left side of the figure shows the interaction transcript (not available to the user at runtime). The right side of the figure shows the NM which will be discussed in the next section.

Our system behaves as follows. First, based on the analysis of the user essay, it selects a question segment to correct misconceptions or to elicit more complete explanations. Next the system asks every question from this question segment. If the user answer is correct, the system simply moves on to the next question (e.g. $Tutor_2 \rightarrow Tutor_3$). For incorrect answers there are two alternatives. For simple questions, the system will give out the correct answer accompanied by a short explanation and move on to the next question (e.g. $Tutor_1 \rightarrow Tutor_2$). For complex questions (e.g. applying physics laws), ITSPOKE will engage into a *remediation subdialogue* that attempts to remediate user's lack of knowledge or skills (e.g. $Tutor_4 \rightarrow Tutor_5$). The remediation subdialogue for each complex question is specified in another question segment.

Our system exhibits some of the issues we linked in Section 1 with complex-domain systems. Dialogues with our system can be long and complex (e.g. the question segment hierarchical structure can reach level 6) and sometimes the system's turn can be quite long (e.g. $Tutor_2$). User's reduced knowledge of the task is also inherent in tutoring.

3 The Navigation Map (NM)

We use the Grosz & Sidner theory of discourse (Grosz and Sidner, 1986) to inform our NM design. According to this theory, each discourse has a discourse purpose/intention. Satisfying the main discourse purpose is achieved by satisfying several smaller purposes/intentions organized in a hierarchical structure. As a result, the discourse is segmented into discourse segments each with an associated discourse segment purpose/intention. This theory has inspired several generic dialogue managers for spoken dialogue systems (e.g. (Rich and Sidner, 1998)).

The NM requires that we have the discourse structure information at runtime. To do that, we manually annotate the system's internal representation of the tutoring task with discourse segment purpose and hierarchy information. Based on this annotation, we can easily construct the discourse structure at runtime. In this section we describe our annotation and the NM design choices we made. Figure 1 shows the state of the NM after turn $Tutor_5$ as the user sees it on the interface (NM line numbering is for exposition only). Note that Figure 1 is not a screenshot of the actual system interface. The NM is the only part from the actual system interface. Figure 2 shows the NM after turn $Tutor_1$.

We manually annotated each system question/explanation for its intention(s)/purpose(s). Note that some system turns have multiple inten-

TUTOR$_1$: To analyze this problem we will first describe the motion of the person and his keys while he is holding them. Then we will look at the motion of the person and his keys after he lets go of them.

Let's begin by looking at the motion of the man and his keys while he is holding them.

How does his velocity compare to that of his keys?

User answer$_1$: man's velocity is bigger [incorrect]

TUTOR$_2$: Well… The problem statement says that the man is holding his keys motionless in front of his face. That means they are both falling together and are not moving with respect to each other. Hence their velocities are the same.

Now let's see what happens when he releases them.

To do so we will first find the individual forces exerted on the man and his keys, then both net forces, both accelerations, and finally we will describe and compare the velocities.

So what are the forces exerted on the man after he releases his keys? Please, specify their directions.

User answer$_2$: gravity vertically down [correct]

TUTOR$_3$: Good. Because there is just one force, gravity's, it is trivial to determine the NET force. So what is the direction of the NET force on the person?

User answer$_3$: down [correct]

TUTOR$_4$: Ok. Considering the net force, does the person have an acceleration? If yes, specify its direction.

User answer$_4$: the man has no acceleration [incorrect]

TUTOR$_5$: Well… To analyze the man's acceleration we will use Newton's second law. What is the definition of this law?

***Problem**: Suppose a man is in a free-falling elevator and is holding his keys motionless right in front of his face. He then lets go. What will be the position of the keys relative to the man's face as time passes? Explain.*

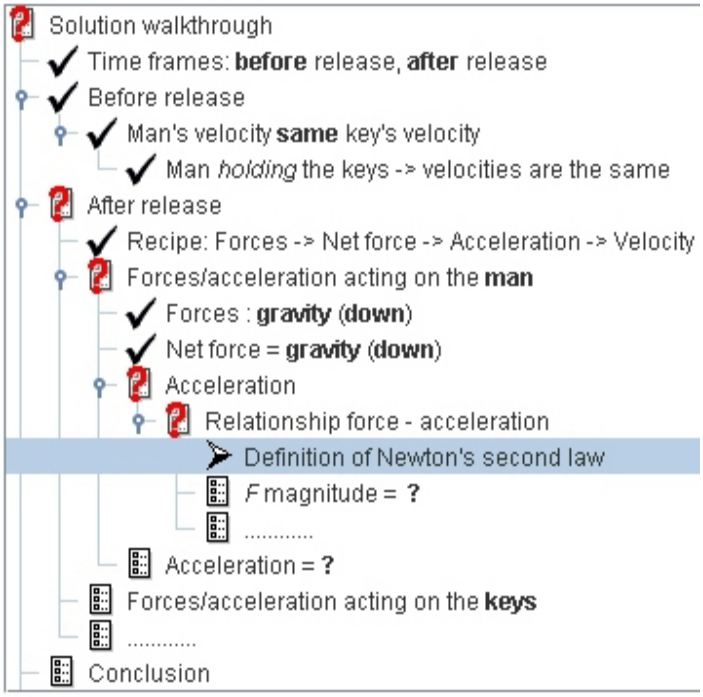


Figure 1. Transcript of a sample ITSPOKE speech interaction (left). The NM as the user sees it after turn Tutor$_5$

tions/purposes thus multiple discourse segments were created for them. For example, in Tutor$_1$ the system first identifies the time frames on which the analysis will be performed (Figure 1&2, NM$_2$). Next, the system indicates that it will discuss about the first time frame (Figure 1&2, NM$_3$) and then it asks the actual question (Figure 2, NM$_4$).

Thus, in addition to our manual annotation of the discourse segment purpose information, we manually organized all discourse segments from a question segment in a hierarchical structure that reflects the discourse structure.

At runtime, while discussing a question segment, the system has only to follow the annotated hierarchy, displaying and highlighting the discourse segment purposes associated with the uttered content. For example, while uttering Tutor$_1$, the NM will synchronously highlight NM$_2$, NM$_3$ and NM$_4$. Remediation question segments (e.g. NM$_{12}$) or explanations (e.g. NM$_5$) activated by incorrect answers are attached to the structure under the corresponding discourse segment.

3.1 NM Design Choices

In our graphical representation of the discourse structure, we used a left to right indented layout. In addition, we made several design choices to enrich the NM information content and usability.

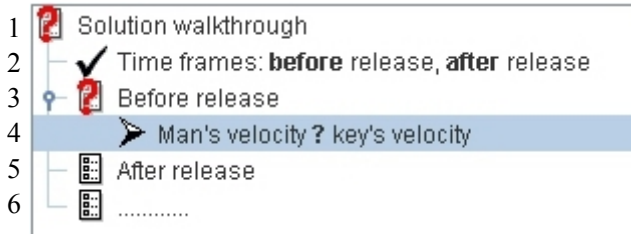


Figure 2. NM state after turn Tutor$_1$

Correct answers. In Figure 2 we show the state of the NM after uttering Tutor$_1$. The current discourse segment purpose (NM$_4$) indicates that the system is asking about the relationship between the two velocities. While we could have kept the same information after the system was done with this discourse segment, we thought that users will benefit from having the correct answer on the screen (recall NM$_4$ in Figure 1). Thus, the NM was enhanced to display the correct answer after the system is done with each question. We extracted the correct answer from the system specifications for each question and manually created a new version of the discourse segment purpose that includes this information.

Limited horizon. Since in our case the system drives the conversation (i.e. system initiative), we always know what questions would be discussed next. We hypothesized that by having access to this information, users will have a better idea of where instruction is heading, thus facilitating their understanding of the relevance of the current topic to the overall discussion. To prevent information overload, we only display the next discourse segment purpose at each level in the hierarchy (see Figure 1, NM_{14}, NM_{16}, NM_{17} and NM_{19}; Figure 2, NM_5); additional discourse segments at the same level are signaled through a dotted line. To avoid helping the students answer the current question in cases when the next discourse segment hints/describes the answer, each discourse segment has an additional purpose annotation that is displayed when the segment is part of the visible horizon.

Auto-collapse. To reduce the amount of information on the screen, discourse segments discussed in the past are automatically collapsed by the system. For example, in Figure 1, NM Line 3 is collapsed in the actual system and Lines 4 and 5 are hidden (shown in Figure1 to illustrate our discourse structure annotation.). The user can expand nodes as desired using the mouse.

Information highlight. Bold and italics font were used to highlight important information (what and when to highlight was manually annotated). For example, in Figure 1, NM_2 highlights the two time frames as they are key steps in approaching this problem. Correct answers are also highlighted.

We would like to reiterate that the goal of this study is to investigate if making certain types of discourse information explicitly available to the user provides any benefits. Thus, whether we have made the optimal design choices is of secondary importance. While, we believe that our annotation is relatively robust as the system questions follow a carefully designed tutoring plan, in the future we would like to investigate these issues.

4 User Study

We designed a user study focused primarily on user's perception of the NM presence/absence. We used a within-subject design where each user received instruction both with and without the NM.

Each user went through the same experimental procedure: 1) read a short document of background material, 2) took a pretest to measure initial physics knowledge, 3) worked through 2 problems with ITSPOKE 4) took a posttest similar to the pretest, 5) took a **NM survey**, and 6) went through a brief open-question interview with the experimenter.

In the 3rd step, the NM was enabled in *only one* problem. Note that in both problems, users did not have access to the system turn transcript. After each problem users filled in a **system questionnaire** in which they rated the system on various dimensions; these ratings were designed to cover dimensions the NM might affect (see Section 5.1). While the system questionnaire implicitly probed the NM utility, the NM survey from the 5th step explicitly asked the users whether the NM was useful and on what dimensions (see Section 5.1)

To account for the effect of the tutored problem on the user's questionnaire ratings, users were randomly assigned to one of two conditions. The users in the first condition (**F**) had the NM enabled in the first problem and disabled in the second problem, while users in the second condition (**S**) had the opposite. Thus, if the NM has any effect on the user's perception of the system, we should see a *decrease* in the questionnaire ratings from problem 1 to problem 2 for *F* users and an *increase* for *S* users.

Other factors can also influence our measurements. To reduce the effect of the text-to-speech component, we used a version of the system with human prerecorded prompts. We also had to account for the amount of instruction as in our system the top level question segment is tailored to what users write in the essay. Thus the essay analysis component was disabled; for all users, the system started with the same top level question segment which assumed no information in the essay. Note that the actual dialogue depends on the correctness of the user answers. After the dialogue, users were asked to revise their essay and then the system moved on to the next problem.

The collected corpus comes from 28 users (13 in *F* and 15 in *S*). The conditions were balanced for gender (*F*: 6 male, 7 female; *S*: 8 male, 7 female). There was no significant differences between the two conditions in terms of pretest ($p<0.63$); in both conditions users learned (significant difference between pretest and posttest, $p<0.01$).

5 Results

5.1 Subjective metrics

Our main resource for investigating the effect of the NM was the system questionnaires given after

each problem. These questionnaires are identical and include 16 questions that probed user's perception of ITSPOKE on various dimensions. Users were asked to answer the questions on a scale from 1-5 (1 – Strongly Disagree, 2 – Disagree, 3 – Somewhat Agree, 4 – Agree, 5 – Strongly Agree). If indeed the NM has any effect we should observe differences between the ratings of the NM problem and the noNM problem (i.e. the NM is disabled).

Table 1 lists the 16 questions in the questionnaire order. The table shows for every question the average rating for all condition-problem combinations (e.g. column 5: condition *F* problem 1 with the NM enabled). For all questions except Q7 and Q11 a higher rating is better. For Q7 and Q11 (italicized in Table 1) a *lower* rating is better as they gauge negative factors (high level of concentration and task disorientation). They also served as a deterrent for negligence while rating.

To test if the NM presence has a significant effect, a repeated-measure ANOVA with between-subjects factors was applied. The within-subjects factor was the NM presence (**NMPres**) and the between-subjects factor was the condition (**Cond**)[1]. The significance of the effect of each factor and their combination (NMPres*Cond) is listed in the table with significant and trend effects highlighted in bold (see columns 2-4). Post-hoc t-tests between the NM and noNM ratings were run for each condition ("s"/"t"marks significant/trend differences).

Results for Q1-6

Questions Q1-6 were inspired by previous work on spoken dialogue system evaluation (e.g. (Walker et al., 2000)) and measure user's overall perception of the system. We find that the NM presence significantly improves user's perception of the system in terms of their ability to concentrate on the instruction (Q3), in terms of their inclination to reuse the system (Q6) and in terms of the system's matching of their expectations (Q4). There is a trend that it was easier for them to learn from the NM enabled version of the system (Q2).

Results for Q7-13

Q7-13 relate directly to our hypothesis that users benefit from access to the discourse structure information. These questions probe the user's perception of ITSPOKE during the dialogue. We find that for 6 out 7 questions the NM presence has a significant/trend effect (Table 1, column 2).

Structure. Users perceive the system as having a structured tutoring plan significantly[2] more in the NM problems (Q8). Moreover, it is significantly easier for them to follow this tutoring plan if the NM is present (Q11). These effects are very clear for *F* users where their ratings differ significantly between the first (NM) and the second problem (noNM). A difference in ratings is present for *S* users but it is not significant. As with most of the *S* users' ratings, we believe that the NM presentation order is responsible for the mostly non-significant differences. More specifically, assuming that the NM has a positive effect, the *S* users are asked to rate first the poorer version of the system (noNM) and then the better version (NM). In contrast, *F* users' task is easier as they already have a high reference point (NM) and it is easier for them to criticize the second problem (noNM). Other factors that can blur the effect of the NM are domain learning and user's adaptation to the system.

Integration. Q9 and Q10 look at how well users think they integrate the system questions in both a forward-looking fashion (Q9) and a backward looking fashion (Q10). Users think that it is significantly easier for them to integrate the current system question to what will be discussed in the future if the NM is present (Q9). Also, if the NM is present, it is easier for users to integrate the current question to the discussion so far (Q10, trend). For Q10, there is no difference for *F* users but a significant one for *S* users. We hypothesize that domain learning is involved here: *F* users learn better from the first problem (NM) and thus have less issues solving the second problem (noNM). In contrast, *S* users have more difficulties in the first problem (noNM), but the presence of the NM eases their task in the second problem.

Correctness. The correct answer NM feature is useful for users too. There is a trend that it is easier for users to know the correct answer if the NM is present (Q13). We hypothesize that speech recognition and language understanding errors are re-

[1] Since in this version of ANOVA the NM/noNM ratings come from two different problems based on the condition, we also run an ANOVA in which the within-subjects factor was the problem (Prob). In this case, the NM effect corresponds to an effect from Prob*Cond which is identical in significance with that of NMPres.

[2] We refer to the significance of the NMPres factor (Table 1, column 2). When discussing individual experimental conditions, we refer to the post-hoc t-tests.

Question	ANOVA			Average rating: F condition				S condition			
	NMPres	Cond	NMPres* Cond	P1		P2		P2		P1	
Overall				NM		noNM		NM		noNM	
1. The tutor increased my understanding of the subject	0.518	0.898	0.862	4.0	>	3.9		4.0	>	3.9	
2. It was easy to learn from the tutor	**0.100**	0.813	0.947	3.9	>	3.6		3.9	>	3.5	
3. The tutor helped me to concentrate	**0.016**	0.156	0.854	3.5	>	3.0		3.9	$>^t$	3.4	
4. The tutor worked the way I expected it to	**0.034**	0.886	0.157	3.5	>	3.4		3.9	$>^s$	3.1	
5. I enjoyed working with the tutor	0.154	0.513	0.917	3.5	>	3.2		3.7	>	3.4	
6. Based on my experience using the tutor to learn physics, I would like to use such a tutor regularly	**0.004**	0.693	0.988	3.7	$>^s$	3.2		3.5	$>^s$	3.0	
During the conversation with the tutor:											
7. ... a high level of concentration is required to follow the tutor	**0.004**	0.534	0.545	3.5	$<^s$	4.2		3.9	$<^t$	4.3	
8. ... the tutor had a clear and structured agenda behind its explanations	**0.008**	0.340	0.104	4.4	$>^s$	3.6		4.3	>	4.1	
9. ... it was easy to figure out where the tutor's instruction was leading me	**0.017**	0.472	0.593	4.0	$>^s$	3.4		4.1	>	3.7	
10. ... when the tutor asked me a question I knew why it was asking me that question	**0.054**	0.191	**0.054**	3.5	~	3.5		4.3	$>^s$	3.5	
11. ... it was easy to loose track of where I was in the interaction with the tutor	**0.012**	0.766	**0.048**	2.5	$<^s$	3.5		2.9	<	3.0	
12. ... I knew whether my answer to the tutor's question was correct or incorrect	0.358	0.635	0.804	3.5	>	3.3		3.7	>	3.4	
13. ... whenever I answered incorrectly, it was easy to know the correct answer after the tutor corrected me	**0.085**	**0.044**	0.817	3.8	>	3.5		4.3	>	3.9	
At the end of the conversation with the tutor:											
14. ... it was easy to understand the tutor's main point	**0.071**	**0.056**	0.894	4.0	>	3.6		4.4	>	4.1	
15. ... I knew what was wrong or missing from my essay	0.340	0.965	0.340	3.9	~	3.9		3.7	<	4.0	
16. ... I knew how to modify my essay	0.791	0.478	0.327	4.1	>	3.9		3.7	<	3.8	

Table 1. System questionnaire results

sponsible for the non-significant NM effect on the dimension captured by Q12.

Concentration. Users also think that the NM enabled version of the system requires less effort in terms of concentration (Q7). We believe that having the discourse segment purpose as visual input allows the users to concentrate more easily on what the system is uttering. In many of the open question interviews users stated that it was easier for them to listen to the system when they had the discourse segment purpose displayed on the screen.

Results for Q14-16

Questions Q14-16 were included to probe user's post tutoring perceptions. We find a trend that in the NM problems it was easier for users to understand the system's main point (Q14). However, in terms of identifying (Q15) and correcting (Q16) problems in their essay the results are inconclusive. We believe that this is due to the fact that the essay interpretation component was disabled in this experiment. As a result, the instruction did not match the initial essay quality. Nonetheless, in the open-question interviews, many users indicated using the NM as a reference while updating their essay.

In addition to the 16 questions, in the system questionnaire after the second problem users were asked to choose which version of the system they preferred the most (i.e. the first or the second problem version). 24 out 28 users (86%) preferred the NM enabled version. In the open-question interview, the 4 users that preferred the noNM version (2 in each condition) indicated that it was harder for them to concurrently concentrate on the audio and the visual input (divided attention problem) and/or that the NM was changing too fast.

To further strengthen our conclusions from the system questionnaire analysis, we would like to note that users were not asked to directly compare the two versions but they were asked to individually rate two versions which is a noisier process (e.g. users need to recall their previous ratings).

The NM survey

While the system questionnaires probed users' NM usage indirectly, in the second to last step in the experiments, users had to fill a NM survey

which explicitly asked how the NM helped them, if at all. The answers were on the same 1 to 5 scale. We find that the majority of users (75%-86%) agreed or strongly agreed that the NM helped them follow the dialogue, learn more easily, concentrate and update the essay. These findings are on par with those from the system questionnaire analysis.

5.2 Objective metrics

Our analysis of the subjective user evaluations shows that users think that the NM is helpful. We would like to see if this perceived usefulness is reflected in any objective metrics of performance. Due to how our experiment was designed, the effect of the NM can be reliably measured only in the first problem as in the second problem the NM is toggled[3]; for the same reason, we can not use the pretest/posttest information.

Our preliminary investigation [4] found several dimensions on which the two conditions differed in the first problem (*F* users had NM, *S* users did not). We find that if the NM was present the interaction was shorter on average and users gave more correct answers; however these differences are not statistically significant (Table 2). In terms of speech recognition performance, we looked at two metrics: AsrMis and SemMis (ASR/Semantic Misrecognition). A user turn is labeled as AsrMis if the output of the speech recognition is different from the human transcript (i.e. a binary version of Word Error Rate). SemMis are AsrMis that change the correctness interpretation. We find that if the NM was present users had fewer AsrMis and fewer SemMis (trend for SemMis, $p<0.09$).

In addition, a χ^2 dependency analysis showed that the NM presence interacts significantly with both AsrMis ($p<0.02$) and SemMis ($p<0.001$), with fewer than expected AsrMis and SemMis in the NM condition. The fact that in the second problem the differences are much smaller (e.g. 2% for AsrMis) and that the NM-AsrMis and NM-SemMis interactions are not significant anymore, suggests that our observations can not be attributed to a difference in population with respect to system's ability to recognize their speech. We hypothesize that these differences are due to the NM text influencing users' lexical choice.

Metric	*F* (NM)	*S* (noNM)	p
# user turns	21.8 (5.3)	22.8 (6.5)	0.65
% correct turns	72% (18%)	67% (22%)	0.59
AsrMis	37% (27%)	46% (28%)	0.46
SemMis	5% (6%)	12% (14%)	0.09

Table 2. Average (standard deviation) for objective metrics in the first problem

6 Related work

Discourse structure has been successfully used in non-interactive settings (e.g. understanding specific lexical and prosodic phenomena (Hirschberg and Nakatani, 1996) , natural language generation (Hovy, 1993), essay scoring (Higgins et al., 2004) as well as in interactive settings (e.g. predictive/generative models of postural shifts (Cassell et al., 2001), generation/interpretation of anaphoric expressions (Allen et al., 2001), performance modeling (Rotaru and Litman, 2006)).

In this paper, we study the utility of the discourse structure on the user side of a dialogue system. One related study is that of (Rich and Sidner, 1998). Similar to the NM, they use the discourse structure information to display a segmented interaction history (**SIH**): an indented view of the interaction augmented with purpose information. This paper extends over their work in several areas. The most salient difference is that here we investigate the benefits of displaying the discourse structure information for the users. In contrast, (Rich and Sidner, 1998) never test the utility of the SIH. Their system uses a GUI-based interaction (no speech/text input, no speech output) while we look at a speech-based system. Also, their underlying task (air travel domain) is much simpler than our tutoring task. In addition, the SIH is not always available and users have to activate it manually.

Other visual improvements for dialogue-based computer tutors have been explored in the past (e.g. talking heads (Graesser et al., 2003)). However, implementing the NM in a new domain requires little expertise as previous work has shown

[3] Due to random assignment to conditions, before the first problem the *F* and *S* populations are similar (e.g. no difference in pretest); thus any differences in metrics can be attributed to the NM presence/absence. However, in the second problem, the two populations are not similar anymore as they have received different forms of instruction; thus any difference has to be attributed to the NM presence/absence in this problem as well as to the NM absence/presence in the previous problem.

[4] Due to logging issues, 2 *S* users are excluded from this analysis (13 *F* and 13 *S* users remaining). We run the subjective metric analysis from Section 5.1 on this subset and the results are similar.

that naïve users can reliably annotate the information needed for the NM (Passonneau and Litman, 1993). Our NM design choices should also have an equivalent in a new domain (e.g. displaying the recognized user answer can be the equivalent of the correct answers). Other NM usages can also be imagined: e.g. reducing the length of the system turns by removing text information that is implicitly represented in the NM.

7 Conclusions & Future work

In this paper we explore the utility of the Navigation Map, a graphical representation of the discourse structure. As our first step towards understanding the benefits of the NM, we ran a user study to investigate if users perceive the NM as useful. From the users' perspective, the NM presence allows them to better identify and follow the tutoring plan and to better integrate the instruction. It was also easier for users to concentrate and to learn from the system if the NM was present. Our preliminary analysis on objective metrics shows that users' preference for the NM version is reflected in more correct user answers and less speech recognition problems in the NM version.

These findings motivate future work in understanding the effects of the NM. We would like to continue our objective metrics analysis (e.g. see if users are better in the NM condition at updating their essay and at answering questions that require combining facts previously discussed). We also plan to run an additional user study with a between-subjects experimental design geared towards objective metrics. The experiment will have two conditions: NM present/absent for all problems. The conditions will then be compared in terms of various objective metrics. We would also like to know which information sources represented in the NM (e.g. discourse segment purpose, limited horizon, correct answers) has the biggest impact.

Acknowledgements

This work is supported by NSF Grants 0328431 and 0428472. We would like to thank Shimei Pan, Pamela Jordan and the ITSPOKE group.

References

K. Acomb, J. Bloom, K. Dayanidhi, P. Hunter, P. Krogh, E. Levin and R. Pieraccini. 2007. *Technical Support Dialog Systems: Issues, Problems, and Solutions.* In Proc. of Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies.

J. Allen, G. Ferguson, B. N., D. Byron, N. Chambers, M. Dzikovska, L. Galescu and M. Swift. 2006. Chester: Towards a Personal Medication Advisor. *Journal of Biomedical Informatics, 39*(5).

J. Allen, G. Ferguson and A. Stent. 2001. *An architecture for more realistic conversational systems.* In Proc. of Intelligent User Interfaces.

J. Cassell, Y. I. Nakano, T. W. Bickmore, C. L. Sidner and C. Rich. 2001. *Non-Verbal Cues for Discourse Structure.* In Proc. of ACL.

A. Graesser, K. Moreno, J. Marineau, A. Adcock, A. Olney and N. Person. 2003. *AutoTutor improves deep learning of computer literacy: Is it the dialog or the talking head?* In Proc. of Artificial Intelligence in Education (AIED).

B. Grosz and C. L. Sidner. 1986. Attentions, intentions and the structure of discourse. *Computational Linguistics, 12*(3).

D. Higgins, J. Burstein, D. Marcu and C. Gentile. 2004. *Evaluating Multiple Aspects of Coherence in Student Essays.* In Proc. of HLT-NAACL.

J. Hirschberg and C. Nakatani. 1996. *A prosodic analysis of discourse segments in direction-giving monologues.* In Proc. of ACL.

E. Hovy. 1993. Automated discourse generation using discourse structure relations. *Articial Intelligence, 63*(Special Issue on NLP).

D. Litman and S. Silliman. 2004. *ITSPOKE: An intelligent tutoring spoken dialogue system.* In Proc. of HLT/NAACL.

S. Oviatt, R. Coulston and R. Lunsford. 2004. *When Do We Interact Multimodally? Cognitive Load and Multimodal Communication Patterns.* In Proc. of International Conference on Multimodal Interfaces.

R. Passonneau and D. Litman. 1993. *Intention-based segmentation: Human reliability and correlation with linguistic cues.* In Proc. of ACL.

H. Pon-Barry, K. Schultz, E. O. Bratt, B. Clark and S. Peters. 2006. Responding to Student Uncertainty in Spoken Tutorial Dialogue Systems. *International Journal of Artificial Intelligence in Education, 16.*

C. Rich and C. L. Sidner. 1998. COLLAGEN: A Collaboration Manager for Software Interface Agents. *User Modeling and User-Adapted Interaction, 8*(3-4).

M. Rotaru and D. Litman. 2006. *Exploiting Discourse Structure for Spoken Dialogue Performance Analysis.* In Proc. of EMNLP.

M. Walker, D. Litman, C. Kamm and A. Abella. 2000. Towards Developing General Models of Usability with PARADISE. *Natural Language Engineering*.

Automated Vocabulary Acquisition and Interpretation in Multimodal Conversational Systems

Yi Liu **Joyce Y. Chai** **Rong Jin**
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824, USA
{liuyi3, jchai, rongjin}@cse.msu.edu

Abstract

Motivated by psycholinguistic findings that eye gaze is tightly linked to human language production, we developed an unsupervised approach based on translation models to automatically learn the mappings between words and objects on a graphic display during human machine conversation. The experimental results indicate that user eye gaze can provide useful information to establish such mappings, which have important implications in automatically acquiring and interpreting user vocabularies for conversational systems.

1 Introduction

To facilitate effective human machine conversation, it is important for a conversational system to have knowledge about user vocabularies and understand how these vocabularies are mapped to the internal entities for which the system has representations. For example, in a multimodal conversational system that allows users to converse with a graphic interface, the system needs to know what vocabularies users tend to use to describe objects on the graphic display and what (type of) object(s) a user is attending to when a particular word is expressed. Here, we use *acquisition* to refer to the process of acquiring relevant vocabularies describing internal entities, and *interpretation* to refer to the process of automatically identifying internal entities given a particular word. Both acquisition and interpretation have been traditionally approached by either knowledge engineering (e.g., manually created lexicons) or supervised learning from annotated data. In this paper, we describe an unsupervised approach that relies on naturally co-occurred eye gaze and spoken utterances during human machine conversation to automatically acquire and interpret vocabularies.

Motivated by psycholinguistic studies (Just and Carpenter, 1976; Griffin and Bock, 2000; Tenenhaus et al., 1995) and recent investigations on computational models for language acquisition and grounding (Siskind, 1995; Roy and Pentland, 2002; Yu and Ballard, 2004), we are particularly interested in two unique questions related to multimodal conversational systems: (1) In a multimodal conversation that involves more complex tasks (e.g., both user initiated tasks and system initiated tasks), is there a reliable temporal alignment between eye gaze and spoken references so that the coupled inputs can be used for automated vocabulary acquisition and interpretation? (2) If such an alignment exists, how can we model this alignment and automatically acquire and interpret the vocabularies?

To address the first question, we conducted an empirical study to examine the temporal relationships between eye fixations and their corresponding spoken references. As shown later in section 4, although a larger variance (compared to the findings from psycholinguistic studies) exists in terms of how eye gaze is linked to speech production during human machine conversation, eye fixations and the corresponding spoken references still occur in a very close vicinity to each other. This natural coupling between eye gaze and speech provides an opportunity to automatically learn the mappings between

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 368–375,
Prague, Czech Republic, June 2007.

words and objects without any human supervision.

Because of the larger variance, it is difficult to apply rule-based approaches to quantify this alignment. Therefore, to address the second question, we developed an approach based on statistical translation models to explore the co-occurrence patterns between eye fixated objects and spoken references. Our preliminary experiment results indicate that the translation model can reliably capture the mappings between the eye fixated objects and the corresponding spoken references. Given an object, this model can provide possible words describing this object, which represents the acquisition process; given a word, this model can also provide possible objects that are likely to be described, which represents the interpretation process.

In the following sections, we first review some related work and introduce the procedures used to collect eye gaze and speech data during human machine conversation. We then describe our empirical study and the unsupervised approach based on translation models. Finally, we present experiment results and discuss their implications in natural language processing applications.

2 Related Work

Our work is motivated by previous work in the following three areas: psycholinguistics studies, multimodal interactive systems, and computational modeling of language acquisition and grounding.

Previous psycholinguistics studies have shown that the direction of gaze carries information about the focus of the user's attention (Just and Carpenter, 1976). Specifically, in human language processing tasks, eye gaze is tightly linked to language production. The perceived visual context influences spoken word recognition and mediates syntactic processing (Tenenhaus et al., 1995). Additionally, before speaking a word, the eyes usually move to the objects to be mentioned (Griffin and Bock, 2000). These psycholinguistics findings have provided a foundation for our investigation.

In research on multimodal interactive systems, recent work indicates that the speech and gaze integration patterns can be modeled reliably for individual users and therefore be used to improve multimodal system performances (Kaur et al., 2003). Studies have also shown that eye gaze has a potential to improve resolution of underspecified referring expressions in spoken dialog systems (Campana et al., 2001) and to disambiguate speech input (Tanaka, 1999). In contrast to these earlier studies, our work focuses on a different goal of using eye gaze for automated vocabulary acquisition and interpretation.

The third area of research that influenced our work is computational modeling of language acquisition and grounding. Recent studies have shown that multisensory information (e.g., through vision and language processing) can be combined to effectively acquire words to their perceptually grounded objects in the environment (Siskind, 1995; Roy and Pentland, 2002; Yu and Ballard, 2004). Especially in (Yu and Ballard, 2004), an unsupervised approach based on a generative correspondence model was developed to capture the mapping between spoken words and the occurring perceptual features of objects. This approach is most similar to the translation model used in our work. However, compared to this work where multisensory information comes from vision and language processing, our work focuses on a different aspect. Here, instead of applying vision processing on objects, we are interested in eye gaze behavior when users interact with a graphic display. Eye gaze is an implicit and subconscious input modality during human machine interaction. Eye gaze data inevitably contain a significant amount of noise. Therefore, it is the goal of this paper to examine whether this modality can be utilized for vocabulary acquisition for conversational systems.

3 Data Collection

We used a *simplified* multimodal conversational system to collect synchronized speech and eye gaze data. A room interior scene was displayed on a computer screen, as shown in Figure 1. While watching the graphical display, users were asked to communicate with the system on topics about the room decorations. A total of 28 objects (e.g., multiple lamps and picture frames, a bed, two chairs, a candle, a dresser, etc., as marked in Figure 1) are explicitly modeled in this scene. The system is *simplified* in the sense that it only supports 14 tasks during human machine interaction. These tasks are designed to cover both open-ended utterances (e.g., the system

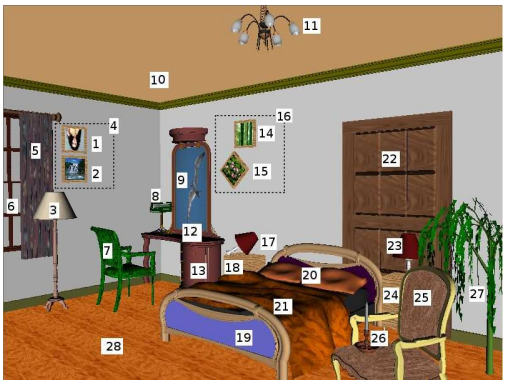


Figure 1: The room interior scene for user studies. For easy reference, we give each object an ID. These IDs are hidden from the system users.

asks users to describe the room) and more restricted utterances (e.g., the system asks the user whether he/she likes the bed) that are commonly supported in conversational systems. Seven human subjects participated in our study.

User speech inputs were recorded using the Audacity software[1], with each utterance time-stamped. Eye movements were recorded using an EyeLink II eye tracker sampled at 250Hz. The eye tracker automatically saved two-dimensional coordinates of a user's eye fixations as well as the time-stamps when the fixations occurred.

The collected raw gaze data is extremely noisy. To refine the gaze data, we further eliminated invalid and saccadic gaze points (known as "saccadic suppression" in vision studies). Since eyes do not stay still but rather make small, frequent jerky movements, we also smoothed the data by averaging nearby gaze locations to identify fixations.

4 Empirical Study on Speech-Gaze Alignment

Based on the data collected, we investigated the temporal alignment between co-occurred eye gaze and spoken utterances. In particular, we examined the temporal alignment between eye gaze fixations and the corresponding spoken references (i.e., the spoken words that are used to refer to the objects on the graphic display).

According to the time-stamp information, we can measure the length of time gap between a user's eye fixation falling on an object and the corresponding spoken reference being uttered (which we refer to as "length of time gap" for brevity). Also, we can count the number of times that user fixations happen to change their target objects during this time gap (which we refer to as "number of fixated object changes" for brevity). The nine most frequently occurred spoken references in utterances from all users (as shown in Table 1) are chosen for this empirical study. For each of those spoken references, we use human judgment to decide which object is referred to. Then, from both before and after the onset of the spoken reference, we find the closest occurrence of the fixation falling on that particular object. Altogether we have 96 such speech-gaze pairs. In 54 pairs, the eye gaze fixation occurred before the corresponding speech reference was uttered; and in the other 42 pairs, the eye fixation occurred after the corresponding speech reference was uttered. This observation suggests that in human machine conversation, eye fixation on an object does not necessarily always proceed the utterance of the corresponding speech reference.

Further, we computed the average *absolute* length of the time gap and the average number of fixated object changes, as well as their variances for each of 5 selected users[2] as shown in Table 1. From Table 1, it is easy to observe that: **(I)** A spoken reference always appears within a short period of time (usually 1-2 seconds) *before or after* the corresponding eye gaze fixation. But, the exact length of the period is far from constant. **(II)** It is not necessary for a user to utter the corresponding spoken reference *immediately* before or after the eye gaze fixation falls on that particular object. Eye gaze fixations may move back and forth. Between the time an object is fixated and the corresponding spoken reference is uttered, a user's eye gaze may fixate on a few other objects (reflected by the average number of eye fixated object changes shown in the table). **(III)** There is a large variance in both the length of time gap and the number of fixated object changes in terms of 1) the same user and the same spoken reference at different time-stamps, 2) the same user but different spo-

[1] http://audacity.sourceforge.net/

[2] The other two users are not selected because the nine selected words do not appear frequently in their utterances.

Spoken	Average Absolute Length of Time Gap (in seconds)					Average Number of Eye Fixated Object Changes				
Reference	User 1	User 2	User 3	User 4	User 5	User 1	User 2	User 3	User 4	User 5
bed	1.27 ± 1.40	1.02 ± 0.65	0.32 ± 0.21	0.59 ± 0.77	2.57 ± 3.25	2.1 ± 3.2	2.1 ± 2.2	0.4 ± 0.5	1.4 ± 2.2	5.3 ± 7.9
tree	-	0.24 ± 0.24	-	-	-	-	0.0 ± 0.0	-	-	-
window	-	0.67 ± 0.74	-	-	1.95 ± 3.20	-	0.0 ± 0.0	-	-	3.3 ± 5.9
mirror	-	1.04 ± 1.36	-	-	-	-	1.0 ± 1.4	-	-	-
candle	-	-	3.64 ± 0.59	-	-	-	-	8.5 ± 2.1	-	-
waterfall	1.80 ± 1.12	-	-	-	-	5.5 ± 4.9	-	-	-	-
painting	0.10 ± 0.10	-	-	-	-	0.2 ± 0.4	-	-	-	-
lamp	0.74 ± 0.54	1.70 ± 0.99	0.26 ± 0.35	1.98 ± 1.72	2.84 ± 2.42	1.3 ± 1.3	1.8 ± 1.5	0.3 ± 0.6	4.8 ± 4.3	2.7 ± 2.2
door	2.47 ± 0.84	-	-	2.49 ± 1.90	6.36 ± 2.29	5.0 ± 2.6	-	-	6.7 ± 5.5	13.3 ± 6.7

Table 1: The average absolute length of time and the number of eye fixated object changes within the time gap of eye gaze and corresponding spoken references. Variances are also listed. Some of the entries are not available because the spoken references were never or rarely used by the corresponding users.

ken references, and 3) the same spoken reference but different users. We believe this is due to the different dialog scenarios and user language habits.

To summarize our empirical study, we find that in human machine conversation, there still exists a natural temporal coupling between user speech and eye gaze, i.e. the spoken reference and the corresponding eye fixation happen within a close vicinity of each other. However, a large variance is also observed in terms of these temporal vicinities, which indicates an intrinsically more complex gaze-speech pattern. Therefore, it is hard to directly quantify the temporal or ordering relationship between spoken references and corresponding eye fixated objects (for example, through rules).

To better handle the complexity in the gaze-speech pattern, we propose to use statistical translation models. Given a time window of enough length, a speech input that contains a list of spoken references (e.g., definite noun phrases) is always accompanied by a list of naturally occurred eye fixations and therefore a list of objects receiving those fixations. All those pairs of speech references and corresponding fixated objects could be viewed as *parallel*, i.e. they *co-occur within the time window*. This situation is very similar to the training process of translation models in statistical machine translation (Brown et al., 1993), where parallel corpus is used to find the mappings between words from different languages by exploiting their co-occurrence patterns. The same idea can be borrowed here: by exploring the co-occurrence statistics, we hope to uncover the exact mapping between those eye fixated objects and spoken references. The intuition is that, the more often a fixation is found to exclusively co-occur with a spoken reference, the more likely a mapping should be established between them.

5 Translation Models for Vocabulary Acquisition and Interpretation

Formally, we denote the set of observations by $\mathbf{D} = \{\mathbf{w}_i, \mathbf{o}_i\}_{i=1}^{N}$ where $\mathbf{w}_i$ and $\mathbf{o}_i$ refers to the i-th speech utterance (i.e., a list of words of spoken references) and the i-th corresponding eye gaze pattern (i.e., a list of eye fixated objects) respectively. When we study the problem of mapping given objects to words (for vocabulary acquisition), the parameter space $\mathbf{\Theta} = \{\Pr(w_j|o_k), 1 \leq j \leq m^w, 1 \leq k \leq m^o\}$ consists of the mapping probabilities of an arbitrary word w_j to an arbitrary object o_k, where m^w and m^o represent the total number of unique words and objects respectively. Those mapping probabilities are subject to constraints $\sum_{j=1}^{m^w} \Pr(w_j|o_k) = 1$. Note that $\Pr(w_j|o_k) = 0$ if the corresponding word w_j and o_k never co-occur in any observed list pair $(\mathbf{w}_i, \mathbf{o}_i)$.

Let l_i^w and l_i^o denote the length of lists $\mathbf{w}_i$ and $\mathbf{o}_i$ respectively. To distinguish with the notations w_j and o_k whose subscripts are indices for *unique* words and objects respectively, we use $\tilde{w}_{i,j}$ to denote the word in the j-th position of the list $\mathbf{w}_i$ and $\tilde{o}_{i,k}$ to denote the object in the k-th position of the list $\mathbf{o}_i$. In translation models, we assume that any word in the list $\mathbf{w}_i$ is mapped to an object in the corresponding list $\mathbf{o}_i$ or a *null object* (we reserve the position 0 for it in every object list). To denote all the word-object mappings in the i-th list pair, we introduce an alignment vector $\mathbf{a}_i$, whose element $a_{i,j}$ takes the value k if the word $\tilde{w}_{i,j}$ is mapped to $\tilde{o}_{i,k}$.

Then, the likelihood of the observations given the

parameters can be computed as follows

$$\Pr(\mathbf{D};\mathbf{\Theta}) = \prod_{i=1}^{N}\Pr(\mathbf{w}_i|\mathbf{o}_i) = \prod_{i=1}^{N}\sum_{\mathbf{a}_i}\Pr(\mathbf{w}_i,\mathbf{a}_i|\mathbf{o}_i)$$
$$= \prod_{i=1}^{N}\sum_{\mathbf{a}_i}\frac{\Pr(l_i^w|\mathbf{o}_i)}{(l_i^o+1)^{l_i^w}}\prod_{j=1}^{l_i^w}\Pr(\tilde{w}_{i,j}|\tilde{o}_{a_{i,j}})$$
$$= \prod_{i=1}^{N}\frac{\Pr(l_i^w|\mathbf{o}_i)}{(l_i^o+1)^{l_i^w}}\sum_{\mathbf{a}_i}\prod_{j=1}^{l_i^w}\Pr(\tilde{w}_{i,j}|\tilde{o}_{a_{i,j}})$$

Note that the following equation holds:

$$\prod_{j=1}^{l_i^w}\sum_{k=0}^{l_i^o}\Pr(\tilde{w}_{i,j}|\tilde{o}_{i,k}) = \sum_{a_{i,1}=1}^{l_i^o}\cdots\sum_{a_{i,l_i^w}=1}^{l_i^o}\prod_{j=1}^{l_i^w}\Pr(\tilde{w}_{i,j}|\tilde{o}_{a_{i,j}})$$

where the right-hand side is actually the expansion of $\sum_{\mathbf{a}_i}\prod_j^{l_i^w}\Pr(\tilde{w}_{i,j}|\tilde{o}_{a_{i,j}})$. Therefore, the likelihood can be simplified as

$$\Pr(\mathbf{D};\mathbf{\Theta}) = \prod_{i=1}^{N}\frac{\Pr(l_i^w|\mathbf{o}_i)}{(l_i^o+1)^{l_i^w}}\prod_{j=1}^{l_i^w}\sum_{k=0}^{l_i^o}\Pr(\tilde{w}_{i,j}|\tilde{o}_{i,k})$$

Switching to the notations w_j and o_k, we have

$$\Pr(\mathbf{D};\mathbf{\Theta}) = \prod_{i=1}^{N}\frac{\Pr(l_i^w|\mathbf{o}_i)}{(l_i^o+1)^{l_i^w}}\prod_{j=1}^{m^w}\left[\sum_{k=0}^{m^o}\Pr(w_j|o_k)\delta_{i,k}^o\right]^{\delta_{i,j}^w}$$

where $\delta_{i,j}^w = 1$ if $\tilde{w}_{i,j} \in \mathbf{w}_i$ and $\delta_{i,j}^w = 0$ otherwise, and $\delta_{i,k}^o = 1$ if $\tilde{o}_{i,k} \in \mathbf{o}_i$ and $\delta_{i,k}^o = 0$ otherwise.

Finally, the translation model can be formalized as the following optimization problem

$$\begin{aligned}\arg\max_{\mathbf{\Theta}} \quad & \log\Pr(\mathbf{D};\mathbf{\Theta}) \\ s.t. \quad & \sum_{j=1}^{m^w}\Pr(w_j|o_k) = 1, \forall k\end{aligned}$$

This optimization problem can be solved by the EM algorithm (Brown et al., 1993).

The above model is developed in the context of mapping given objects to words, i.e., its solution yields a set of conditional probabilities $\{\Pr(w_j|o_k), \forall j\}$ for each object o_k, indicating how likely every word is mapped to it. Similarly, we can develop the model in the context of mapping given words to objects (for vocabulary interpretation), whose solution leads to another set of probabilities $\{\Pr(o_k|w_j), \forall k\}$ for each word w_j indicating how likely every object is mapped to it. In our experiments, both models are implemented and we will present the results later.

6 Experiments

We experimented our proposed statistical translation model on the collected data mentioned in Section 3.

6.1 Preprocessing

The main purpose of preprocessing is to create a "parallel corpus" for training a translation model. Here, the "parallel corpus" refers to a series of speech-gaze pairs, each of them consisting of a list of words from the spoken references in the user utterances and a list of objects that are fixated upon within the same time window.

Specifically, we first transcribed the user speech into scripts by automatic speech recognition software and then refined them manually. A time-stamp was associated with each word in the speech script. Further, we detected long pauses in the speech script as splitting points to create time windows, since a long pause usually marks the start of a sentence that indicates a user's attention shift. In our experiment, we set the threshold of judging a long pause to be 1 second. From all the data gathered from 7 users, we get 357 such time windows (which typically contain 10-20 spoken words and 5-10 fixated object changes).

Given a time window, we then found the objects being fixated upon by eye gaze (represented by their IDs as shown in Figure 1). Considering that eye gaze fixation could occur during the pauses in speech, we expanded each time window by a fixed length at both its start and end to find the fixations. In our experiments, the expansion length is set to 0.5 seconds.

Finally, we applied a part-of-speech tagger to each sentence in the user script and only singled out nouns as potential spoken references in the word list. The Porter stemming algorithm was also used to get the normalized forms of those nouns.

The translation model was trained based on this preprocessed parallel data.

6.2 Evaluation Metrics

As described in Section 5, by using a statistical translation model we can get a set of translation probabilities, either from any given spoken word to all the objects, or from any given object to all the spoken words. To evaluate the two sets of translation probabilities, we use *precision* and *recall* as

#Rank	Precision	Recall	#Rank	Precision	Recall
1	0.6667	0.2593	6	0.2302	0.5370
2	0.4524	0.3519	7	0.2041	0.5556
3	0.3810	0.4444	8	0.1905	0.5926
4	0.3095	0.4815	9	0.1799	0.6296
5	0.2667	0.5185	10	0.1619	0.6296

Table 2: Average precision/recall of mapping given objects to words (i.e., acquisition)

#Rank	Precision	Recall	#Rank	Precision	Recall
1	0.7826	0.3214	6	0.3043	0.7500
2	0.5870	0.4821	7	0.2671	0.7679
3	0.4638	0.5714	8	0.2446	0.8036
4	0.3804	0.6250	9	0.2293	0.8393
5	0.3478	0.7143	10	0.2124	0.8571

Table 3: Average precision/recall of mapping given words to objects.(i.e., interpretation)

evaluation metrics.

Specifically, for a given object o_k the translation model will yield a set of probabilities $\{\Pr(w_j|o_k), \forall j\}$. We can sort the probabilities and get a ranked list. Let us assume that we have the ground truth about all the spoken words to which the given object should be mapped. Then, at a given number n of top ranked words, the *precision* of mapping the given object o_k to words is defined as

$$\frac{\text{\# words that } o_k \text{ is correctly mapped to}}{\text{\# words that } o_k \text{ is mapped to}}$$

and the *recall* is defined as

$$\frac{\text{\# words that } o_k \text{ is correctly mapped to}}{\text{\# words that } o_k \text{ should be mapped to}}$$

All the counting above is done within the top n rank. Therefore, we can get different precision/recall at different ranks. At each rank, the overall performance can be evaluated by averaging the precision/recall for all the given objects. Human judgment is used to decide whether an object-word mapping is correct or not, as ground truth for evaluation.

Similarly, based on the set of probabilities of mapping a given object with spoken words, we can find a ranked list of objects for a given word, i.e. $\{\Pr(o_k|w_j), \forall k\}$. Thus, at a given rank the *precision* and *recall* of mapping a given word w_j to objects can be measured.

6.3 Experiment Results

Vocabulary acquisition is the process of finding the appropriate word(s) for any given object. For the sake of statistical significance, our evaluation is done on 21 objects that were mentioned at least 3 times by the users.

Table 2 gives the average precision/recall evaluated at the top 10 ranks. As we can see, if we use the most probable word acquired for each object, about 66.67% of them are appropriate. With the rank increasing, more and more appropriate words can be acquired. About 62.96% of all the appropriate words are included within the top 10 probable words found. The results indicate that by using a translation model, we can obtain the words that are used by the users to describe the objects with reasonable accuracy.

Table 4 presents the top 3 most probable words found for each object. It shows that although there may be more than one word appropriate to describe a given object, those words with highest probabilities always suggest the most popular way of describing the corresponding object among the users. For example, for the object with ID 26, the word `candle` gets a higher probability than the word `candlestick`, which is in accordance with our observation that in our user study, on most occasions users tend to use the word `candle` rather than the word `candlestick`.

Vocabulary interpretation is the process of finding the appropriate object(s) for any given spoken word. Out of 176 nouns in the user vocabulary, we only evaluate those used at least three times for statistical significance concerns. Further, abstract words (such as `reason, position`) and general words (such as `room, furniture`) are not evaluated since they do not refer to any particular objects in the scene. Finally, 23 nouns remain for evaluation.

We manually enumerated all the object(s) that those 23 nouns refer to as the ground truth in our evaluation. Note that a given noun can possibly be used to refer to multiple objects, such as `lamp`, since we have several lamps (with object ID 3, 8, 17, and 23) in the experiment setting, and `bed`, since bed frame, bed spread, and pillows (with object ID 19, 21, and 20 respectively) are all part of a bed. Also, an object can be referred to by multiple nouns. For example, the words `painting, picture`, or `waterfall` can all be used to refer to the object with ID 15.

Object	Rank 1	Rank 2	Rank 3
1	paint (0.254) *	wall (0.191)	left (0.150)
2	pictur (0.305) *	girl (0.122)	niagara (0.095) *
3	wall (0.109)	lamp (0.093) *	floor (0.084)
4	upsid (0.174) *	left (0.151) *	paint (0.149) *
5	pictur (0.172)	window (0.157) *	wall (0.116)
6	window (0.287) *	curtain (0.115)	pictur (0.076)
7	chair (0.287) *	tabl (0.088)	bird (0.083)
9	mirror (0.161) *	dresser (0.137)	bird (0.098) *
12	room (0.131)	lamp (0.127)	left (0.069)
14	hang (0.104)	favourit (0.085)	natur (0.064)
15	thing (0.066)	size (0.059)	queen (0.057)
16	paint (0.211) *	pictur (0.116) *	forest (0.076) *
17	lamp (0.354) *	end (0.154)	tabl (0.097)
18	bedroom (0.158)	side (0.128)	bed (0.104)
19	bed (0.576) *	room (0.059)	candl (0.049)
20	bed (0.396) *	queen (0.211) *	size (0.176)
21	bed (0.180) *	chair (0.097)	orang (0.078)
22	bed (0.282)	door (0.235) *	chair (0.128)
25	chair (0.215) *	bed (0.162)	candlestick (0.124)
26	candl (0.145) *	chair (0.114)	candlestick (0.092) *
27	tree (0.246) *	chair (0.107)	floor (0.096)

Table 4: Words found for given objects. Each row lists the top 3 most probable spoken words (being stemmed) for the corresponding given object, with the mapping probabilities in parentheses. Asterisks indicate correctly identified spoken words. Note that some objects are heavily overlapped, so the corresponding words are considered correct for all the overlapping objects, such as bed being considered correct for objects with ID 19, 20, and 21.

Word	Rank 1	Rank 2	Rank 3	Rank 4
curtain	6 (0.305) *	5 (0.305) *	7 (0.133)	1 (0.121)
candlestick	25 (0.147) *	28 (0.135)	24 (0.131)	22 (0.117)
lamp	22 (0.126)	12 (0.094)	17 (0.093) *	25 (0.093)
dresser	12 (0.298) *	9 (0.294) *	13 (0.173) *	7 (0.104)
queen	20 (0.187) *	21 (0.182) *	22 (0.136)	19 (0.136) *
door	22 (0.200) *	27 (0.124)	25 (0.108)	24 (0.106)
tabl	9 (0.152) *	12 (0.125) *	13 (0.112) *	22 (0.107)
mirror	9 (0.251) *	12 (0.238)	8 (0.109)	13 (0.081)
girl	2 (0.173)	22 (0.128)	16 (0.099)	10 (0.074)
chair	22 (0.132)	25 (0.099) *	28 (0.085)	24 (0.082)
waterfal	6 (0.226)	5 (0.215)	1 (0.118)	9 (0.083)
candl	19 (0.156)	22 (0.139)	28 (0.134)	24 (0.131)
niagara	4 (0.359) *	2 (0.262) *	1 (0.226)	7 (0.045)
plant	27 (0.230) *	22 (0.181)	23 (0.131)	28 (0.117)
tree	27 (0.352) *	22 (0.218)	26 (0.100)	13 (0.062)
upsid	4 (0.204) *	12 (0.188)	9 (0.153)	1 (0.104) *
bird	9 (0.142) *	10 (0.138)	12 (0.131)	7 (0.121)
desk	12 (0.170) *	9 (0.141) *	19 (0.118)	8 (0.118)
bed	19 (0.207) *	22 (0.141)	20 (0.111) *	28 (0.090)
upsidedown	4 (0.243) *	3 (0.219)	6 (0.203)	5 (0.188)
paint	4 (0.188) *	16 (0.148) *	1 (0.137) *	15 (0.118) *
window	6 (0.305) *	5 (0.290) *	3 (0.085)	22 (0.065)
lampshad	3 (0.223) *	7 (0.137)	11 (0.137)	10 (0.137)

Table 5: Objects found for given words. Each row lists the 4 most probable object IDs for the corresponding given words (being stemmed), with the mapping probabilities in parentheses. Asterisks indicate correctly identified objects. Note that some objects are heavily overlapped, such as the candle (with object ID 26) and the chair (with object ID 25), and both were considered correct for the respective spoken words.

Table 3 gives the average precision/recall evaluated at the top 10 ranks. As we can see, if we use the most probable object found for each speech word, about 78.26% of them are appropriate. With the rank increasing, more and more appropriate objects can be found. About 85.71% of all the appropriate objects are included within the top 10 probable objects found. The results indicate that by using a translation model, we can predict the objects from user spoken words with reasonable accuracy.

Table 5 lists the top 4 probable objects found for each spoken word being evaluated. A close look reveals that in general, the top ranked objects tend to gather around the correct object for a given spoken word. This is consistent with the fact that eye gaze tends to move back and forth. It also indicates that the mappings established by the translation model can effectively find the approximate area of the corresponding fixated object, even if it cannot find the object due to the noisy and jerky nature of eye gaze.

The precision/recall in vocabulary acquisition is not as high as that in vocabulary interpretation, partially due to the relatively small scale of our experiment data. For example, with only 7 users' speech data on 14 conversational tasks, some words were only spoken a few times to refer to an object, which prevented them from getting a significant portion of probability mass among all the words in the vocabulary. This degrades both precision and recall. We believe that in large scale experiments or real-world applications, the performance will be improved.

7 Discussion and Conclusion

Previous psycholinguistic findings have shown that eye gaze is tightly linked with human language production. During human machine conversation, our study shows that although a larger variance is observed on how eye fixations are exactly linked with corresponding spoken references (compared to the psycholinguistic findings), eye gaze in general is closely coupled with corresponding referring expressions in the utterances. This close coupling nature between eye gaze and speech utterances provides an opportunity for the system to automatically

acquire different words related to different objects without any human supervision. To further explore this idea, we developed a novel unsupervised approach using statistical translation models.

Our experimental results have shown that this approach can reasonably uncover the mappings between words and objects on the graphical display. The main advantages of this approach include: 1) It is an unsupervised approach with minimum human inference; 2) It does not need any prior knowledge to train a statistical translation model; 3) It yields probabilities that indicate the reliability of the mappings.

Certainly, our current approach is built upon simplified assumptions. It is quite challenging to incorporate eye gaze information since it is extremely noisy with large variances. Recent work has shown that the effect of eye gaze in facilitating spoken language processing varies among different users (Qu and Chai, 2007). In addition, visual properties of the interface also affect user gaze behavior and thus influence the predication of attention (Prasov et al., 2007) based on eye gaze. Our future work will develop models to address these variations.

Nevertheless, the results from our current work have several important implications in building robust conversational interfaces. First of all, most conversational systems are built with static knowledge space (e.g., vocabularies) and can only be updated by the system developers. Our approach can potentially allow the system to automatically acquire knowledge and vocabularies based on the natural interactions with the users without human intervention. Furthermore, the automatically acquired mappings between words and objects can also help language interpretation tasks such as reference resolution. Given the recent advances in eye tracking technology (Duchowski, 2002), integrating non-intrusive and high performance eye trackers with conversational interfaces becomes feasible. The work reported here can potentially be integrated in practical systems to improve the overall robustness of human machine conversation.

Acknowledgment

This work was supported by funding from National Science Foundation (IIS-0347548, IIS-0535112, and IIS-0643494) and Disruptive Technology Office. The authors would like to thank Zahar Prasov for his contribution to data collection.

References

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

E. Campana, J. Baldridge, J. Dowding, B. A. Hockey, R. Remington, and L. S. Stone. 2001. Using eye movements to determine referents in a spoken dialog system. In *Proceedings of PUI'01*.

A. T. Duchowski. 2002. A breath-first survey of eye tracking applications. *Behavior Research methods, Instruments, and Computers*, 33(4).

Z. M. Griffin and K. Bock. 2000. What the eyes say about speaking. *Psychological Science*, 11:274–279.

M. A. Just and P. A. Carpenter. 1976. Eye fixations and cognitive processes. *Cognitive Psychology*, 8:441–480.

M. Kaur, M. Tremaine, N. Huang, J. Wilder, Z. Gacovski, F. Flippo, and C. S. Mantravadi. 2003. Where is "it"? Event synchronization in gaze-speech input systems. In *Proceedings of ICMI'03*, pages 151–157.

Z. Prasov, J. Y. Chai, and H. Jeong. 2007. Eye gaze for attention prediction in multimodal human-machine conversation. In *2007 Spring Symposium on Interaction Challenges for Artificial Assistants*, Palo Alto, California, March.

S. Qu and J. Y. Chai. 2007. An exploration of eye gaze in spoken language processing for multimodal conversational interfaces. In *NAACL'07*, pages 284–291, Rochester, New York, April.

D. Roy and A. Pentland. 2002. Learning words from sights and sounds, a computational model. *Cognitive Science*, 26(1):113–1146.

J. M. Siskind. 1995. Grounding language in perception. *Artificial Intelligence Review*, 8:371–391.

K. Tanaka. 1999. A robust selection system using real-time multi-modal user-agent interactions. In *Proceedings of IUI'99*, pages 105–108.

M. K. Tenenhaus, M. Sivey-Knowlton, E. Eberhard, and J. Sedivy. 1995. Integration of visual and linguistic information during spoken language comprehension. *Science*, 268:1632–1634.

C. Yu and D. H. Ballard. 2004. On the integration of grounding language and learning objects. *Proceedings of AAAI'04*.

A Multimodal Interface for Access to Content in the Home

Michael Johnston
AT&T Labs Research,
Florham Park,
New Jersey, USA
johnston@research.att.com

Luis Fernando D'Haro
Universidad Politécnica de Madrid,
Madrid, Spain
lfdharo@die.upm.es

Michelle Levine
AT&T Labs Research,
Florham Park,
New Jersey, USA
mfl@research.att.com

Bernard Renger
AT&T Labs Research,
Florham Park,
New Jersey, USA
renger@research.att.com

Abstract

In order to effectively access the rapidly increasing range of media content available in the home, new kinds of more natural interfaces are needed. In this paper, we explore the application of multimodal interface technologies to searching and browsing a database of movies. The resulting system allows users to access movies using speech, pen, remote control, and dynamic combinations of these modalities. An experimental evaluation, with more than 40 users, is presented contrasting two variants of the system: one combining speech with traditional remote control input and a second where the user has a tablet display supporting speech and pen input.

1 Introduction

As traditional entertainment channels and the internet converge through the advent of technologies such as broadband access, movies-on-demand, and streaming video, an increasingly large range of content is available to consumers in the home. However, to benefit from this new wealth of content, users need to be able to rapidly and easily find what they are actually interested in, and do so effortlessly while relaxing on the couch in their living room — a location where they typically do not have easy access to the keyboard, mouse, and close-up screen display typical of desktop web browsing.

Current interfaces to cable and satellite television services typically use direct manipulation of a graphical user interface using a remote control. In order to find content, users generally have to either navigate a complex, pre-defined, and often deeply embedded menu structure or type in titles or other key phrases using an onscreen keyboard or triple tap input on a remote control keypad. These interfaces are cumbersome and do not scale well as the range of content available increases (Berglund, 2004; Mitchell, 1999).

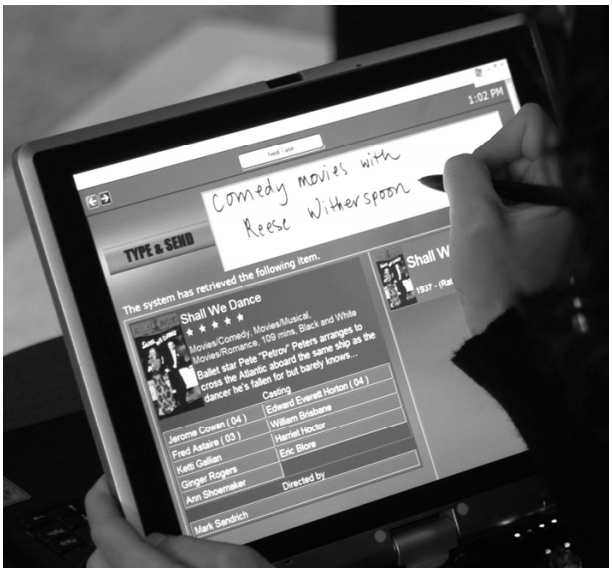


Figure 1 Multimodal interface on tablet

In this paper we explore the application of multimodal interface technologies (See André (2002) for an overview) to the creation of more effective systems used to search and browse for entertainment content in the home. A number of previous systems have investigated the addition of unimodal spoken search queries to a graphical electronic program guide (Ibrahim and Johansson, 2002

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 376–383,
Prague, Czech Republic, June 2007.

(NokiaTV); Goto et al., 2003; Wittenburg et al., 2006). Wittenburg et al experiment with unrestricted speech input for electronic program guide search, and use a highlighting mechanism to provide feedback to the user regarding the "relevant" terms the system understood and used to make the query. However, their usability study results show this complex output can be confusing to users and does not correspond to user expectations. Others have gone beyond unimodal speech input and added multimodal commands combining speech with pointing (Johansson, 2003; Portele et al, 2006). Johansson (2003) describes a movie recommender system MadFilm where users can use speech and pointing to accept/reject recommended movies. Portele et al (2006) describe the SmartKom-Home system which includes multimodal electronic program guide on a tablet device.

In our work we explore a broader range of interaction modalities and devices. The system provides users with the flexibility to interact using spoken commands, handwritten commands, unimodal pointing (GUI) commands, and multimodal commands combining speech with one or more pointing gestures made on a display. We compare two different interaction scenarios. The first utilizes a traditional remote control for direct manipulation and pointing, integrated with a wireless microphone for speech input. In this case, the only screen is the main TV display (far screen). In the second scenario, the user also has a second graphical display (close screen) presented on a mobile tablet which supports speech and pen input, including both pointing and handwriting (Figure 1). Our application task also differs, focusing on search and browsing of a large database of movies-on-demand and supporting queries over multiple simultaneous dimensions. This work also differs in the scope of the evaluation. Prior studies have primarily conducted qualitative evaluation with small groups of users (5 or 6). A quantitative and qualitative evaluation was conducted examining the interaction of 44 naïve users with two variants of the system. We believe this to be the first broad scale experimental evaluation of a flexible multimodal interface for searching and browsing large databases of movie content.

In Section 2, we describe the interface and illustrate the capabilities of the system. In Section 3, we describe the underlying multimodal processing architecture and how it processes and integrates user inputs. Section 4 describes our experimental evaluation and comparison of the two systems. Section 5 concludes the paper.

2 Interacting with the system

The system described here is an advanced user interface prototype which provides multimodal access to databases of media content such as movies or television programming. The current database is harvested from publicly accessible web sources and contains over 2000 popular movie titles along with associated metadata such as cast, genre, director, plot, ratings, length, etc.

The user interacts through a graphical interface augmented with speech, pen, and remote control input modalities. The remote control can be used to move the current focus and select items. The pen can be used both for selecting items (pointing at them) and for handwritten input. The graphical user interface has three main screens. The main screen is the search screen (Figure 2). There is also a control screen used for setting system parameters and a third comparison display used for showing movie details side by side (Figure 4). The user can select among the screens using three icons in the navigation bar at the top left of the screen. The arrows provide 'Back' and 'Next' for navigation through previous searches. Directly below, there is a feedback window which indicates whether the system is listening and provides feedback on speech recognition and search. In the tablet variant, the microphone and speech recognizer are activated by tapping on 'CLICK TO SPEAK' with the pen. In the remote control version, the recognizer can also be activated using a button on the remote control. The main section of the search display (Figure 2) contains two panels. The right panel (results panel) presents a scrollable list of thumbnails for the movies retrieved by the current search. The left panel (details panel) provides details on the currently selected title in the results panel. These include the genre, plot summary, cast, and director.

The system supports a speech modality, a handwriting modality, pointing (unimodal GUI) modality, and composite multimodal input where the user utters a spoken command which is combined with pointing 'gestures' the user has made towards screen icons using the pen or the remote control.

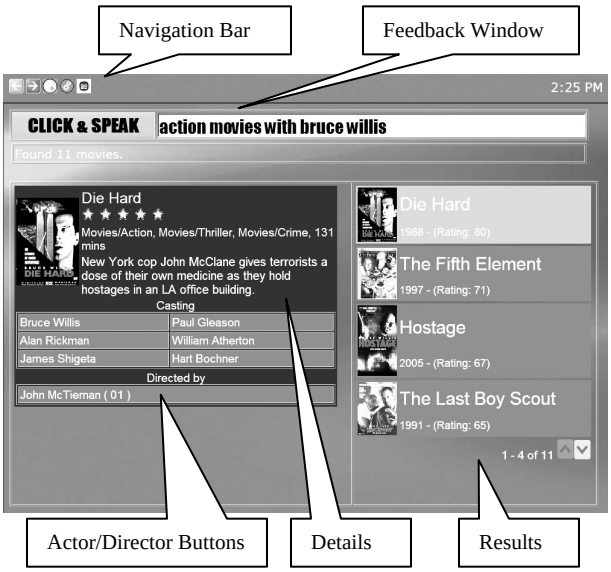


Figure 2 Graphical user interface

Speech: The system supports speech search over multiple different dimensions such as title, genre, cast, director, and year. Input can be more telegraphic with searches such as "Legally Blonde", "Romantic comedy", and "Reese Witherspoon", or more verbose natural language queries such as "I'm looking for a movie called Legally Blonde" and "Do you have romantic comedies". An important advantage of speech is that it makes it easy to combine multiple constraints over multiple dimensions within a single query (Cohen, 1992). For example, queries can indicate co-stars: "movies starring Ginger Rogers and Fred Astaire", or constrain genre and cast or director at the same time: "Meg Ryan Comedies", "show drama directed by Woody Allen" and "show comedy movies directed by Woody Allen and starring Mira Sorvino".

Handwriting: Handwritten pen input can also be used to make queries. When the user's pen approaches the feedback window, it expands allowing for freeform pen input. In the example in Figure 3, the user requests comedy movies with Bruce Willis using unimodal handwritten input. This is an important input modality as it is not impacted by ambient noise such as crosstalk from other viewers or currently playing content.

Figure 3 Handwritten query

Pointing/GUI: In addition to the recognition-based modalities, speech and handwriting, the interface also supports more traditional graphical user interface (GUI) commands. In the details panel, the actors and directors are presented as buttons. Pointing at (i.e., clicking on) these buttons results in a search for all of the movies with that particular actor or director, allowing users to quickly navigate from an actor or director in a specific title to other material they may be interested in. The buttons in the results panel can be pointed at (clicked on) in order to view the details in the left panel for that particular title.

Figure 4 Comparison screen

Composite multimodal input: The system also supports true composite multimodality when spoken or handwritten commands are integrated with pointing gestures made using the pen (in the tablet version) or by selecting items (in the remote control version). This allows users to quickly execute more complex commands by combining the ease of reference of pointing with the expressiveness of spoken constraints. While by unimodally pointing at an actor button you can search for all of the actor's movies, by adding speech you can narrow the search to, for example, all of their comedies by saying: "show comedy movies with THIS actor". Multimodal commands with multiple pointing gestures are also supported, allowing the user to 'glue' together references to multiple actors or directors in order to constrain the search. For example, they can say "movies with THIS actor and THIS director" and point at the 'Alan Rickman' button and then the 'John McTiernan' button in turn (Figure 2). Comparison commands can also be multimo-

dal; for example, if the user says "compare THIS movie and THIS movie" and clicks on the two buttons on the right display for 'Die Hard' and the 'The Fifth Element' (Figure 2), the resulting display shows the two movies side-by-side in the comparison screen (Figure 4).

3 Underlying multimodal architecture

The system consists of a series of components which communicate through a facilitator component (Figure 5). This develops and extends upon the multimodal architecture underlying the MATCH system (Johnston et al., 2002).

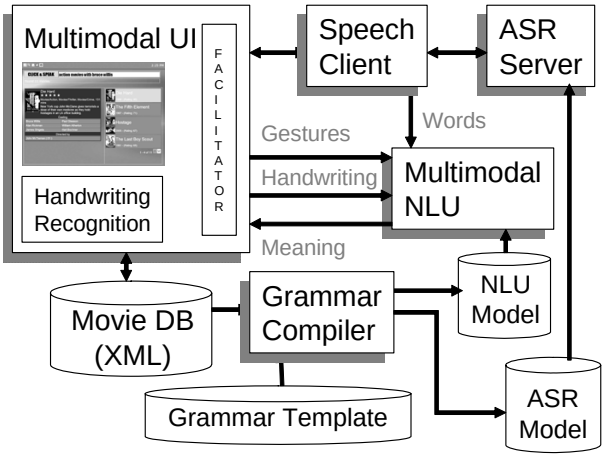


Figure 5 System architecture

The underlying database of movie information is stored in XML format. When a new database is available, a Grammar Compiler component extracts and normalizes the relevant fields from the database. These are used in conjunction with a predefined multimodal grammar template and any available corpus training data to build a multimodal understanding model and speech recognition language model.

The user interacts with the multimodal user interface client (Multimodal UI), which provides the graphical display. When the user presses 'CLICK TO SPEAK' a message is sent to the Speech Client, which activates the microphone and ships audio to a speech recognition server. Handwritten inputs are processed by a handwriting recognizer embedded within the multimodal user interface client. Speech recognition results, pointing gestures made on the display, and handwritten inputs, are all passed to a multimodal understanding server which uses finite-state multimodal language processing techniques (Johnston and Bangalore, 2005) to interpret and integrate the speech and gesture. This model combines alignment of multimodal inputs, multimodal integration, and language understanding within a single mechanism. The resulting combined meaning representation (represented in XML) is passed back to the multimodal user interface client, which translates the understanding results into an XPATH query and runs it against the movie database to determine the new series of results. The graphical display is then updated to represent the latest query.

The system first attempts to find an exact match in the database for all of the search terms in the user's query. If this returns no results, a back off and query relaxation strategy is employed. First the system tries a search for movies that have all of the search terms, except stop words, independent of the order (an AND query). If this fails, then it backs off further to an OR query of the search terms and uses an edit machine, using Levenshtein distance, to retrieve the most similar item to the one requested by the user.

4 Evaluation

After designing and implementing our initial prototype system, we conducted an extensive multimodal data collection and usability study with the two different interaction scenarios: tablet versus remote control. Our main goals for the data collection and statistical analysis were three-fold: collect a large corpus of natural multimodal dialogue for this media selection task, investigate whether future systems should be paired with a remote control or tablet-like device, and determine which types of search and input modalities are more or less desirable.

4.1 Experimental set up

The system evaluation took place in a conference room set up to resemble a living room (Figure 6). The system was projected on a large screen across the room from a couch.

An adjacent conference room was used for data collection (Figure 7). Data was collected in sound files, videotapes, and text logs. Each subject's spoken utterances were recorded by three microphones: wireless, array and stand alone. The wireless microphone was connected to the system while the array and stand alone microphones were

around 10 feet away.[1] Test sessions were recorded with two video cameras – one captured the system's screen using a scan converter while the other recorded the user and couch area. Lastly, the user's interactions and the state of the system were captured by the system's logger. The logger is an additional agent added to the system architecture for the purposes of the evaluation. It receives log messages from different system components as interaction unfolds and stores them in a detailed XML log file. For the specific purposes of this evaluation, each log file contains: general information about the system's components, a description and timestamp for each system event and user event, names and timestamps for the system-recorded sound files, and timestamps for the start and end of each scenario.

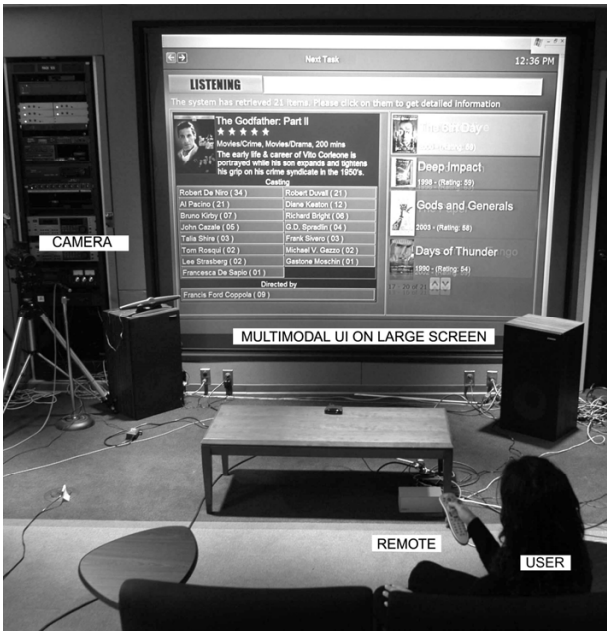


Figure 6 Data collection environment

Forty-four subjects volunteered to participate in this evaluation. There were 33 males and 11 females, ranging from 20 to 66 years of age. Each user interacted with both the remote control and tablet variants of the system, completing the same two sets of scenarios and then freely interacting with each system. For counterbalancing purposes, half of the subjects used the tablet and then the remote control and the other half used the remote control and then the tablet. The scenario set assigned to each version was also counterbalanced.

Figure 7 Data collection room

Each set of scenarios consisted of seven defined tasks, four user-specialized tasks and five open-ended tasks. Defined tasks were presented in chart form and had an exact answer, such as the movie title that two specified actors/actresses starred in. For example, users had to find the movie in the database with Matthew Broderick and Denzel Washington. User-specialized tasks relied on the specific user's preferences, such as "What type of movie do you like to watch on a Sunday evening? Find an example from that genre and write down the title". Open-ended tasks prompted users to search for any type of information with any input modality. The tasks in the two sets paralleled each other. For example, if one set of tasks asked the user to find the highest ranked comedy movie with Reese Witherspoon, the other set of tasks asked the user to find the highest ranked comedy movie with Will Smith. Within each task set, the defined tasks appeared first, then the user-specialized tasks and lastly the open-ended tasks. However, for each participant, the order of defined tasks was randomized, as well as the order of user-specialized tasks.

At the beginning of the session, users read a short tutorial about the system's GUI, the experiment, and available input modalities. Before interacting with each version, users were given a manual on operating the tablet/remote control. To minimize bias, the manuals gave only a general overview with few examples and during the experiment users were alone in the room.

At the end of each session, users completed a user-satisfaction/preference questionnaire and then a qualitative interview. The questionnaire consisted

[1] Here we report results for the wireless microphone only. Analysis of the other microphone conditions is ongoing.

of 25 statements about the system in general, the two variants of the system, input modality options and search options. For example, statements ranged from "If I had [the system], I would use the tablet with it" to "If my spoken request was misunderstood, I would want to try again with speaking". Users responded to each statement with a 5-point Likert scale, where 1 = 'I strongly agree', 2 = 'I mostly agree', 3 = 'I can't say one way or the other', 4 = 'I mostly do not agree' and 5 = 'I do not agree at all'. The qualitative interview allowed for more open-ended responses, where users could discuss reasons for their preferences and their likes and dislikes regarding the system.

4.2 Results

Data was collected from all 44 participants. Due to technical problems, five participants' logs or sound files were not recorded in parts of the experiment. All collected data was used for the overall statistics but these five participants had to be excluded from analyses comparing remote control to tablet.

Spoken utterances: After removing empty sound files, the full speech corpus consists of 3280 spoken utterances. Excluding the five participants subject to technical problems, the total is 3116 utterances (1770 with the remote control and 1346 with the tablet).

The set of 3280 utterances averages 3.09 words per utterance. There was not a significant difference in utterance length between the remote control and tablet conditions. Users' averaged 2.97 words per utterance with the remote control and 3.16 words per utterance with the tablet, paired t (38) = 1.182, p = n.s. However, users spoke significantly more often with the remote control. On average, users spoke 34.51 times with the tablet and 45.38 times with the remote control, paired t (38) = -3.921, $p < .01$.

ASR performance: Over the full corpus of 3280 speech inputs, word accuracy was 44% and sentence accuracy 38%. In the tablet condition, word accuracy averaged 46% and sentence accuracy 41%. In the remote control condition, word accuracy averaged 41% and sentence accuracy 38%. The difference across conditions was only significant for word accuracy, paired t (38) = 2.469, $p < .02$. In considering the ASR performance, it is important to note that 55% of the 3280 speech inputs were out of grammar, and perhaps more importantly 34% were out of the functionality of the system entirely. On within functionality inputs, word accuracy is 62% and sentence accuracy 57%. On the in grammar inputs, word accuracy is 86% and sentence accuracy 83%. The vocabulary size was 3851 for this task. In the corpus, there are a total of 356 out-of-vocabulary words.

Handwriting recognition: Performance was determined by manual inspection of screen capture video recordings.[2] There were a total of 384 handwritten requests with overall 66% sentence accuracy and 76% word accuracy.

Task completion: Since participants had to record the task answers on a paper form, task completion was calculated by whether participants wrote down the correct answer. Overall, users had little difficulty completing the tasks. On average, participants completed 11.08 out of the 14 defined tasks and 7.37 out of the 8 user-specialized tasks. The number of tasks completed did not differ across system variants.[3] For the seven defined tasks within each condition, users averaged 5.69 with the remote control and 5.40 with the tablet, paired t (34) = -1.203, p = n.s. For the four user-specialized task within each condition, users averaged 3.74 on the remote control and 3.54 on the tablet, paired t (34) = -1.268, p = n.s.

Input modality preference: During the interview, 55% of users reported preferring the pointing (GUI) input modality over speech and multimodal input. When asked about handwriting, most users were hesitant to place it on the list. They also discussed how speech was extremely important, and given a system with a low error speech recognizer, using speech for input probably would be their first choice. In the questionnaire, the majority of users (93%) 'strongly agree' or 'mostly agree' with the importance of making a pointing request. The importance of making a request by speaking had the next highest average, where 57% 'strongly agree' or 'mostly agree' with the statement. The importance of multimodal and handwriting requests had the lowest averages, where 39% agreed with the former and 25% for the latter. However, in the open-ended interview, users mentioned handwriting as an important back-up input choice for cases when the speech recognizer fails.

[2] One of the 44 participants videotape did not record and so is not included in the statistics.

[3] Four participants did not properly record their task answers and had to be eliminated from the 39 participants being used in the remote control versus tablet statistics.

Further support for input modality preference was gathered from the log files, which showed that participants mostly searched using unimodal speech commands and GUI buttons. Out of a total of 6082 user inputs to the systems, 48% were unimodal speech and 39% were unimodal GUI (pointing and clicking). Participants requested information with composite multimodal commands 7% of the time and with handwriting 6% of the time.

Search preference: Users most strongly agreed with movie title being the most important way to search. For searching by title, more than half the users chose 'strongly agree' and 91% of users chose 'strongly agree' or 'mostly agree'. Slightly more than half chose 'strongly agree' with searching by actor/actress and slightly less than half chose 'strongly agree' with the importance of searching by genre. During the open ended interview, most users reported title as the most important means for searching.

Variant preference: Results from the qualitative interview indicate that 67% of users preferred the remote control over the tablet variant of the system. The most common reported reasons were familiarity, physical comfort and ease of use. Remote control preference is further supported from the user-preference questionnaire, where 68% of participants 'mostly agree' or 'strongly agree' with wanting to use the remote control variant of the system, compared to 30% of participants choosing 'mostly agree' or 'strongly agree' with wanting to use the tablet version of the system.

5 Conclusion

With the range of entertainment content available to consumers in their homes rapidly expanding, the current access paradigm of direct manipulation of complex graphical menus and onscreen keyboards, and remote controls with way too many buttons is increasingly ineffective and cumbersome. In order to address this problem, we have developed a highly flexible multimodal interface that allows users to search for content using speech, handwriting, pointing (using pen or remote control), and dynamic multimodal combinations of input modes. Results are presented in a straightforward graphical interface similar to those found in current systems but with the addition of icons for actors and directors that can be used both for unimodal GUI and multimodal commands. The system allows users to search for movies over multiple different dimensions of classification (title, genre, cast, director, year) using the mode or modes of their choice. We have presented the initial results of an extensive multimodal data collection and usability study with the system.

Users in the study were able to successfully use speech in order to conduct searches. Almost half of their inputs were unimodal speech (48%) and the majority of users strongly agreed with the importance of using speech as an input modality for this task. However, as also reported in previous work (Wittenburg et al 2006), recognition accuracy remains a serious problem. To understand the performance of speech recognition here, detailed error analysis is important. The overall word accuracy was 44% but the majority of errors resulted from requests from users that lay outside the functionality of the underlying system, involving capabilities the system did not have or titles/cast absent from the database (34% of the 3280 spoken and multimodal inputs). No amount of speech and language processing can resolve these problems. This highlights the importance of providing more detailed help and tutorial mechanisms in order to appropriately ground users' understanding of system capabilities. Of the remaining 66% of inputs (2166) which were within the functionality of the system, 68% were in grammar. On the within functionality portion of the data, the word accuracy was 62%, and on in grammar inputs it is 86%. Since this was our initial data collection, an un-weighted finite-state recognition model was used. The performance will be improved by training stochastic language models as data become available and employing robust understanding techniques. One interesting issue in this domain concerns recognition of items that lie outside of the current database. Ideally the system would have a far larger vocabulary than the current database so that it would be able to recognize items that are outside the database. This would allow feedback to the user to differentiate between lack of results due to recognition or understanding problems versus lack of items in the database. This has to be balanced against degradation in accuracy resulting from increasing the vocabulary.

In practice we found that users, while acknowledging the value of handwriting as a back-up mode, generally preferred the more relaxed and familiar style of interaction with the remote control. However, several factors may be at play here.

The tablet used in the study was the size of a small laptop and because of cabling had a fixed location on one end of the couch. In future, we would like to explore the use of a smaller, more mobile, tablet that would be less obtrusive and more conducive to leaning back on the couch. Another factor is that the in-lab data collection environment is somewhat unrealistic since it lacks the noise and disruptions of many living rooms. It remains to be seen whether in a more realistic environment we might see more use of handwritten input. Another factor here is familiarity. It may be that users have more familiarity with the concept of speech input than handwriting. Familiarity also appears to play a role in user preferences for remote control versus tablet. While the tablet has additional capabilities such handwriting and easier use of multimodal commands, the remote control is more familiar to users and allows for a more relaxed interaction since they can lean back on the couch. Also many users are concerned about the quality of their handwriting and may avoid this input mode for that reason.

Another finding is that it is important not to underestimate the importance of GUI input. 39% of user commands were unimodal GUI (pointing) commands and 55% of users reported a preference for GUI over speech and handwriting for input. Clearly, the way forward for work in this area is to determine the optimal way to combine more traditional graphical interaction techniques with the more conversational style of spoken interaction.

Most users employed the composite multimodal commands, but they make up a relatively small proportion of the overall number of user inputs in the study data (7%). Several users commented that they did not know enough about the multimodal commands and that they might have made more use of them if they had understood them better. This, along with the large number of inputs that were out of functionality, emphasizes the need for more detailed tutorial and online help facilities. The fact that all users were novices with the system may also be a factor. In future, we hope to conduct a longer term study with repeat users to see how previous experience influences use of newer kinds of inputs such as multimodal and handwriting.

Acknowledgements Thanks to Keith Bauer, Simon Byers, Harry Chang, Rich Cox, David Gibbon, Mazin Gilbert, Stephan Kanthak, Zhu Liu, Antonio Moreno, and Behzad Shahraray for their help and support. Thanks also to the Dirección General de Universidades e Investigación - Consejería de Educación - Comunidad de Madrid, España for sponsoring D'Haro's visit to AT&T.

References

Elisabeth André. 2002. Natural Language in Multimodal and Multimedia systems. In Ruslan Mitkov (ed.) *Oxford Handbook of Computational Linguistics*. Oxford University Press.

Aseel Berglund. 2004. *Augmenting the Remote Control: Studies in Complex Information Navigation for Digital TV*. Linköping Studies in Science and Technology, Dissertation no. 872. Linköping University.

Philip R. Cohen. 1992. The Role of Natural Language in a Multimodal Interface. In *Proceedings of ACM UIST Symposium on User Interface Software and Technology*. pp. 143-149.

Jun Goto, Kazuteru Komine, Yuen-Bae Kim and Noriyoshi Uratan. 2003. A Television Control System based on Spoken Natural Language Dialogue. In *Proceedings of 9th International Conference on Human-Computer Interaction.* pp. 765-768.

Aseel Ibrahim and Pontus Johansson. 2002. Multimodal Dialogue Systems for Interactive TV Applications. In *Proceedings of 4th IEEE International Conference on Multimodal Interfaces*. pp. 117-222.

Pontus Johansson. 2003. MadFilm - a Multimodal Approach to Handle Search and Organization in a Movie Recommendation System. In *Proceedings of the 1st Nordic Symposium on Multimodal Communication.* Helsingör, Denmark. pp. 53-65.

Michael Johnston, Srinivas Bangalore, Guna Vasireddy, Amanda Stent, Patrick Ehlen, Marilyn Walker, Steve Whittaker, Preetam Maloor. 2002. MATCH: An Architecture for Multimodal Dialogue Systems. In *Proceedings of the 40th ACL*. pp. 376-383.

Michael Johnston and Srinivas Bangalore. 2005. Finite-state Multimodal Integration and Understanding. *Journal of Natural Language Engineering 11.2*. Cambridge University Press. pp. 159-187.

Russ Mitchell. 1999. TV's Next Episode. *U.S. News and World Report*. 5/10/99.

Thomas Portele, Silke Goronzy, Martin Emele, Andreas Kellner, Sunna Torge, and Jüergen te Vrugt. 2006. SmartKom–Home: The Interface to Home Entertainment. In Wolfgang Wahlster (ed.) *SmartKom: Foundations of Multimodal Dialogue Systems*. Springer. pp. 493-503.

Kent Wittenburg, Tom Lanning, Derek Schwenke, Hal Shubin and Anthony Vetro. 2006. The Prospects for Unrestricted Speech Input for TV Content Search. In *Proceedings of AVI'06*. pp. 352-359.

Fast Unsupervised Incremental Parsing

Yoav Seginer
Institute for Logic, Language and Computation
Universiteit van Amsterdam
Plantage Muidergracht 24
1018TV Amsterdam
The Netherlands
yseginer@science.uva.nl

Abstract

This paper describes an incremental parser and an unsupervised learning algorithm for inducing this parser from plain text. The parser uses a representation for syntactic structure similar to dependency links which is well-suited for incremental parsing. In contrast to previous unsupervised parsers, the parser does not use part-of-speech tags and both learning and parsing are local and fast, requiring no explicit clustering or global optimization. The parser is evaluated by converting its output into equivalent bracketing and improves on previously published results for unsupervised parsing from plain text.

1 Introduction

Grammar induction, the learning of the grammar of a language from unannotated example sentences, has long been of interest to linguists because of its relevance to language acquisition by children. In recent years, interest in unsupervised learning of grammar has also increased among computational linguists, as the difficulty and cost of constructing annotated corpora led researchers to look for ways to train parsers on unannotated text. This can either be semi-supervised parsing, using both annotated and unannotated data (McClosky et al., 2006) or unsupervised parsing, training entirely on unannotated text.

The past few years have seen considerable improvement in the performance of unsupervised parsers (Klein and Manning, 2002; Klein and Manning, 2004; Bod, 2006a; Bod, 2006b) and, for the first time, unsupervised parsers have been able to improve on the right-branching heuristic for parsing English. All these parsers learn and parse from sequences of part-of-speech tags and select, for each sentence, the binary parse tree which maximizes some objective function. Learning is based on global maximization of this objective function over the whole corpus.

In this paper I present an unsupervised parser from plain text which does not use parts-of-speech. Learning is local and parsing is (locally) greedy. As a result, both learning and parsing are fast. The parser is incremental, using a new link representation for syntactic structure. Incremental parsing was chosen because it considerably restricts the search space for both learning and parsing. The representation the parser uses is designed for incremental parsing and allows a prefix of an utterance to be parsed before the full utterance has been read (see section 3). The representation the parser outputs can be converted into bracketing, thus allowing evaluation of the parser on standard treebanks.

To achieve completely unsupervised parsing, standard unsupervised parsers, working from part-of-speech sequences, need first to induce the parts-of-speech for the plain text they need to parse. There are several algorithms for doing so (Schütze, 1995; Clark, 2000), which cluster words into classes based on the most frequent neighbors of each word. This step becomes superfluous in the algorithm I present here: the algorithm collects lists of labels for each word, based on neighboring words, and then directly

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 384–391,
Prague, Czech Republic, June 2007.

uses these labels to parse. No clustering is performed, but due to the Zipfian distribution of words, high frequency words dominate these lists and parsing decisions for words of similar distribution are guided by the same labels.

Section 2 describes the syntactic representation used, section 3 describes the general parser algorithm and sections 4 and 5 complete the details by describing the learning algorithm, the lexicon it constructs and the way the parser uses this lexicon. Section 6 gives experimental results.

2 Common Cover Links

The representation of syntactic structure which I introduce in this paper is based on links between pairs of words. Given an utterance and a bracketing of that utterance, *shortest common cover link sets* for the bracketing are defined. The original bracketing can be reconstructed from any of these link sets.

2.1 Basic Definitions

An *utterance* is a sequence of words $\langle x_1, \ldots, x_n \rangle$ and a *bracket* is any sub-sequence $\langle x_i, \ldots, x_j \rangle$ of consecutive words in the utterance. A set $\mathcal{B}$ of brackets over an utterance U is a *bracketing* of U if every word in U is in some bracket and for any $X, Y \in \mathcal{B}$ either $X \cap Y = \emptyset$, $X \subseteq Y$ or $Y \subseteq X$ (non-crossing brackets). The *depth* of a word $x \in U$ under a bracket $B \in \mathcal{B}$ ($x \in B$) is the maximal number of brackets $X_1, \ldots, X_n \in \mathcal{B}$ such that $x \in X_1 \subset \ldots \subset X_n \subset B$. A word x is a *generator* of depth d of B in $\mathcal{B}$ if x is of minimal depth under B (among all words in B) and that depth is d. A bracket may have more than one generator.

2.2 Common Cover Link Sets

A *common cover link* over an utterance U is a triple $x \xrightarrow{d} y$ where $x, y \in U$, $x \neq y$ and d is a non-negative integer. The word x is the *base* of the link, the word y is its *head* and d is the *depth* of the link. The common cover link set $R_{\mathcal{B}}$ associated with a bracketing $\mathcal{B}$ is the set of common cover links over U such that $x \xrightarrow{d} y \in R_{\mathcal{B}}$ iff the word x is a generator of depth d of the smallest bracket $B \in \mathcal{B}$ such that $x, y \in B$ (see figure 1(a)).

Given $R_{\mathcal{B}}$, a simple algorithm reconstructs the bracketing $\mathcal{B}$: for each word x and depth $0 \leq d$, create a bracket covering x and all y such that for some $d' \leq d$, $x \xrightarrow{d'} y \in R_{\mathcal{B}}$.

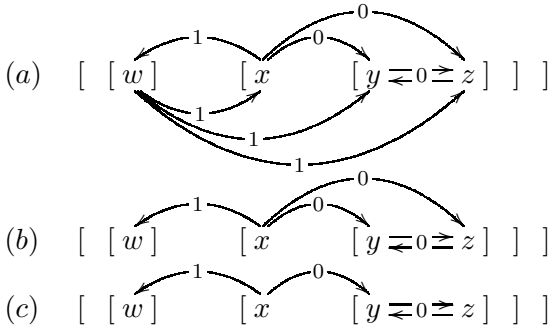


Figure 1: (a) The common cover link set $R_{\mathcal{B}}$ of a bracketing $\mathcal{B}$, (b) a representative subset R of $R_{\mathcal{B}}$, (c) the shortest common cover link set based on R.

Some of the links in the common cover link set $R_{\mathcal{B}}$ are redundant. The first redundancy is the result of brackets having more than one generator. The bracketing reconstruction algorithm outlined above can construct a bracket from the links based at any of its generators. The bracketing $\mathcal{B}$ can therefore be reconstructed from a subset $R \subseteq R_{\mathcal{B}}$ if, for every bracket $B \in \mathcal{B}$, R contains the links based at least at one generator[1] of B. Such a set R is a *representative subset* of $R_{\mathcal{B}}$ (see figure 1(b)).

A second redundancy in the set $R_{\mathcal{B}}$ follows from the *linear transitivity* of $R_{\mathcal{B}}$:

Lemma 1 *If y is between x and z, $x \xrightarrow{d_1} y \in R_{\mathcal{B}}$ and $y \xrightarrow{d_2} z \in R_{\mathcal{B}}$ then $x \xrightarrow{d} z \in R_{\mathcal{B}}$ where if there is a link $y \xrightarrow{d'} x \in R_{\mathcal{B}}$ then $d = \max(d_1, d_2)$ and $d = d_1$ otherwise.*

This property implies that longer links can be deduced from shorter links. It is, therefore, sufficient to leave only the shortest necessary links in the set. Given a representative subset R of $R_{\mathcal{B}}$, a *shortest common cover link set* of $R_{\mathcal{B}}$ is constructed by removing any link which can be deduced from shorter links by linear transitivity. For each representative subset $R \subseteq R_{\mathcal{B}}$, this defines a unique shortest common cover link set (see figure 1(c)).

Given a shortest common cover link set S, the bracketing which it represents can be calculated by

[1] From the bracket reconstruction algorithm it can be seen that links of depth 0 may never be dropped.

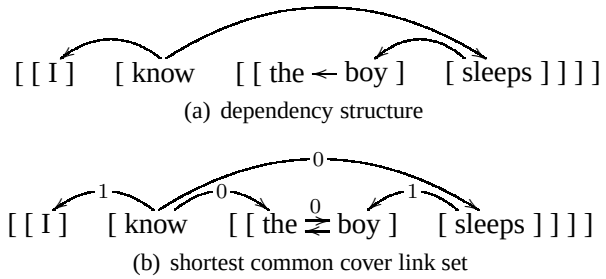


Figure 2: A dependency structure and shortest common cover link set of the same sentence.

first using linear transitivity to deduce missing links and then applying the bracket reconstruction algorithm outlined above for $R_{\mathcal{B}}$.

2.3 Comparison with Dependency Structures

Having defined a link-based representation of syntactic structure, it is natural to wonder what the relation is between this representation and standard dependency structures. The main differences between the two representations can all be seen in figure 2. The first difference is in the linking of the NP *the boy*. While the shortest common cover link set has an exocentric construction for this NP (that is, links going back and forth between the two words), the dependency structure forces us to decide which of the two words in the NP is its head. Considering that linguists have not been able to agree whether it is the determiner or the noun that is the head of an NP, it may be easier for a learning algorithm if it did not have to make such a choice.

The second difference between the structures can be seen in the link from *know* to *sleeps*. In the shortest common cover link set, there is a path of links connecting *know* to each of the words separating it from *sleeps*, while in the dependency structure no such links exist. This property, which I will refer to as *adjacency* plays an important role in incremental parsing, as explained in the next section.

The last main difference between the representations is the assignment of depth to the common cover links. In the present example, this allows us to distinguish between the attachment of the external (subject) and the internal (object) arguments of the verb. Dependencies cannot capture this difference without additional labeling of the links. In what follows, I will restrict common cover links to having depth 0 or 1. This restriction means that any tree represented by a shortest common cover link set will be skewed - every subtree must have a short branch. It seems that this is indeed a property of the syntax of natural languages. Building this restriction into the syntactic representation considerably reduces the search space for both parsing and learning.

3 Incremental Parsing

To calculate a shortest common cover link for an utterance, I will use an incremental parser. Incrementality means that the parser reads the words of the utterance one by one and, as each word is read, the parser is only allowed to add links which have one of their ends at that word. Words which have not yet been read are not available to the parser at this stage. This restriction is inspired by psycholinguistic research which suggests that humans process language incrementally (Crocker et al., 2000). If the incrementality of the parser roughly resembles that of human processing, the result is a significant restriction of parser search space which does not lead to too many parsing errors.

The adjacency property described in the previous section makes shortest common cover link sets especially suitable for incremental parsing. Consider the example given in figure 2. When the word *the* is read, the parser can already construct a link from *know* to *the* without worrying about the continuation of the sentence. This link is part of the correct parse whether the sentence turns out to be *I know the boy* or *I know the boy sleeps*. A dependency parser, on the other hand, cannot make such a decision before the end of the sentence is reached. If the sentence is *I know the boy* then a dependency link has to be created from *know* to *boy* while if the sentence is *I know the boy sleeps* then such a link is wrong. This problem is known in psycholinguistics as the problem of reanalysis (Sturt and Crocker, 1996).

Assume the incremental parser is processing a prefix $\langle x_1, \ldots, x_k \rangle$ of an utterance and has already deduced a set of links L for this prefix. It can now only add links which have one of their ends at x_k and it may never remove any links. From the definitions in section 2.2 it is possible to derive an exact characterization of the links which may be added at each step such that the resulting link set represents some

bracketing. It can be shown that any shortest common cover link set can be constructed incrementally under these conditions. As the full specification of these conditions is beyond the scope of this paper, I will only give the main condition, which is based on adjacency. It states that a link may be added from x to y only if for every z between x and y there is a path of links (in L) from x to z but no link from z to y. In the example in figure 2 this means that when the word *sleeps* is first read, a link to *sleeps* can be created from *know*, *the* and *boy* but not from *I*.

Given these conditions, the parsing process is simple. At each step, the parser calculates a non-negative weight (section 5) for every link which may be added between the prefix $\langle x_1, \ldots, x_{k-1} \rangle$ and x_k. It then adds the link with the strongest positive weight and repeats the process (adding a link can change the set of links which may be added). When all possible links are assigned a zero weight by the parser, the parser reads the next word of the utterance and repeats the process. This is a greedy algorithm which optimizes every step separately.

4 Learning

The weight function which assigns a weight to a candidate link is lexicalized: the weight is calculated based on the lexical entries of the words which are to be connected by the link. It is the task of the learning algorithm to learn the lexicon.

4.1 The Lexicon

The lexicon stores for each word x a lexical entry. Each such lexical entry is a sequence of *adjacency points*, holding statistics relevant to the decision whether to link x to some other word. These statistics are given as weights assigned to labels and linking properties. Each adjacency point describes a different link based at x, similar to the specification of the arguments of a word in dependency parsing.

Let W be the set of words in the corpus. The set of *labels* $L(W) = W \times \{0, 1\}$ consists of two labels based on every word w: a *class label* $(w, 0)$ (denoted by $[w]$) and an *adjacency label* $(w, 1)$ (denoted by $[w\_]$ or $[\_w]$). The two labels $(w, 0)$ and $(w, 1)$ are said to be *opposite labels* and, for $l \in L(W)$, I write l^{-1} for the opposite of l. In addition to the labels, there is also a finite set $P = \{Stop, In^*, In, Out\}$ of *linking properties*. The $Stop$ specifies the strength of non-attachment, In and Out specify the strength of inbound and outbound links and In^* is an intermediate value in the induction of inbound and outbound strengths. A *lexicon* $\mathcal{L}$ is a function which assigns each word $w \in W$ a lexical entry $(\ldots, A^w_{-2}, A^w_{-1}, A^w_1, A^w_2, \ldots)$. Each of the A^w_i is an *adjacency point*.

Each A^w_i is a function $A^w_i : L(W) \cup P \to \mathbb{R}$ which assigns each label in $L(W)$ and each linking property in P a real valued *strength*. For each A^w_i, $\#(A^w_i)$ is the *count* of the adjacency point: the number of times the adjacency point was updated. Based on this count, I also define a normalized version of A^w_i: $\bar{A}^w_i(l) = A^w_i(l)/\#(A^w_i)$.

4.2 The Learning Process

Given a sequence of training utterances $(U_t)_{0 \le t}$, the *learner* constructs a sequence of lexicons $(\mathcal{L}_s)_{0 \le s}$ beginning with the zero lexicon $\mathcal{L}_0$ (which assigns a zero strength to all labels and linking properties). At each step, the learner uses the parsing function $\mathcal{P}_{\mathcal{L}_s}$ based on the previously learned lexicon $\mathcal{L}_s$ to extend the parse L of an utterance U_t. It then uses the result of this parse step (together with the lexicon $\mathcal{L}_s$) to create a new lexicon $\mathcal{L}_{s+1}$ (it may be that $\mathcal{L}_s = \mathcal{L}_{s+1}$). This operation is a *lexicon update*. The process then continues with the new lexicon $\mathcal{L}_{s+1}$. Any of the lexicons $\mathcal{L}_s$ constructed by the learner may be used for parsing any utterance U, but as s increases, parsing accuracy should improve. This learning process is open-ended: additional training text can always be added without having to re-run the learner on previous training data.

4.3 Lexicon Update

To define a lexicon update, I extend the definition of an utterance to be $U = \langle \emptyset_l, x_1, \ldots, x_n, \emptyset_r \rangle$ where $\emptyset_l$ and $\emptyset_r$ are boundary markers. The property of adjacency can now be extended to include the boundary markers. A symbol $\alpha \in U$ is *adjacent* to a word x relative to a set of links L over U if for every word z between x and α there is a path of links in L from x to z but there is no link from z to α. In the following example, the adjacencies of x_1 are $\emptyset_l$, x_2 and x_3:

$$x_1 \text{—}0\text{→} x_2 \qquad x_3 \qquad x_4$$

If a link is added from x_2 to x_3, x_4 becomes adjacent to x_1 instead of x_3 (the adjacencies of x_1 are then $\emptyset_l$, x_2 and x_4):

$$x_1 \text{ —}0\text{→ } x_2 \text{ —}0\text{→ } x_3 \qquad x_4$$

The positions in the utterance adjacent to a word x are indexed by an index i such that $i < 0$ to the left of x, $i > 0$ to the right of x and $|i|$ increases with the distance from x.

The parser may only add a link from a word x to a word y adjacent to x (relative to the set of links already constructed). Therefore, the lexical entry of x should collect statistics about each of the adjacency positions of x. As seen above, adjacency positions may move, so the learner waits until the parser completes parsing the utterance and then updates each adjacency point A_i^x with the symbol α at the ith adjacency position of x (relative to the parse generated by the parser). It should be stressed that this update does not depend on whether a link was created from x to α. In particular, whatever links the parser assigns, $A^x_{(-1)}$ and A^x_1 are always updated by the symbols which appear immediately before and after x.

The following example should clarify the picture. Consider the fragment:

$$\text{put} \text{ —}0\text{→ } \text{the} \rightleftarrows 0 \rightleftarrows \text{box} \qquad \text{on}$$

All the links in this example, including the absence of a link from *box* to *on*, depend on adjacency points of the form $A^x_{(-1)}$ and A^x_1 which are updated independently of any links. Based on this alone and regardless of whether a link is created from *put* to *on*, A_2^{put} will be updated by the word *on*, which is indeed the second argument of the verb *put*.

4.4 Adjacency Point Update

The update of A_i^x by α is given by operations $A_i^x(p)$ += $f(A^\alpha_{(-1)}, A^\alpha_1)$ which make the value of $A_i^x(p)$ in the new lexicon $\mathcal{L}_{s+1}$ equal to the sum $A_i^x(p) + f(A^\alpha_{(-1)}, A^\alpha_1)$ in the old lexicon $\mathcal{L}_s$.

Let $Sign(i)$ be 1 if $0 < i$ and -1 otherwise. Let

$${}_\bullet A_i^\alpha = \begin{cases} true & \text{if } \nexists l \in L(W) : \\ & \quad A_i^\alpha(l) > A_i^\alpha(Stop) \\ false & \text{otherwise} \end{cases}$$

The update of A_i^x by α begins by incrementing the count:

$$\#(A_i^x) \text{ += } 1$$

If α is a boundary symbol ($\emptyset_l$ or $\emptyset_r$) or if x and α are words separated by stopping punctuation (full stop, question mark, exclamation mark, semicolon, comma or dash):

$$A_i^x(Stop) \text{ += } 1$$

Otherwise, for every $l \in L(W)$:

$$A_i^x(l^{-1}) \text{ += } \begin{cases} 1 & \text{if } l = [\alpha] \\ \bar{A}^\alpha_{Sign(-i)}(l) & \text{otherwise} \end{cases}$$

(In practice, only $l = [\alpha]$ and the 10 strongest labels in $A^\alpha_{Sign(-i)}$ are updated. Because of the exponential decay in the strength of labels in $A^\alpha_{Sign(-i)}$, this is a good approximation.)

If $i = -1, 1$ and α is not a boundary or blocked by punctuation, simple bootstrapping takes place by updating the following properties:

$$A_i^x(In^*) \text{ += } \begin{cases} -1 & \text{if } {}_\bullet A^\alpha_{Sign(-i)} \\ +1 & \text{if } \neg {}_\bullet A^\alpha_{Sign(-i)} \wedge {}_\bullet A^\alpha_{Sign(i)} \\ 0 & \text{otherwise} \end{cases}$$

$$A_i^x(Out) \text{ += } \bar{A}^\alpha_{Sign(-i)}(In^*)$$

$$A_i^x(In) \text{ += } \bar{A}^\alpha_{Sign(-i)}(Out)$$

4.5 Discussion

To understand the way the labels and properties are calculated, it is best to look at an example. The following table gives the linking properties and strongest labels for the determiner *the* as learned from the complete Wall Street Journal corpus (only $A^{the}_{(-1)}$ and A^{the}_1 are shown):

the

A_{-1}		A_1	
$Stop$	12897	$Stop$	8
In^*	14898	In^*	18914
In	8625	In	4764
Out	-13184	Out	21922
[the]	10673	[the]	16461
[of_]	6871	[a]	3107
[in_]	5520	[_the]	2787
[a]	3407	[of]	2347
[for_]	2572	[_company]	2094
[to_]	2094	['s]	1686

A strong class label $[w]$ indicates that the word w frequently appears in contexts which are similar to *the*. A strong adjacency label $[w\_]$ (or $[\_w]$) indicates

that w either frequently appears next to *the* or that w frequently appears in the same contexts as words which appear next to *the*.

The property $Stop$ counts the number of times a boundary appeared next to *the*. Because *the* can often appear at the beginning of an utterance but must be followed by a noun or an adjective, it is not surprising that $Stop$ is stronger than any label on the left but weaker than all labels on the right. In general, it is unlikely that a word has an outbound link on the side on which its $Stop$ strength is stronger than that of any label. The opposite is not true: a label stronger than $Stop$ indicates an attachment but this may also be the result of an inbound link, as in the following entry for *to*, where the strong labels on the left are a result of an inbound link:

to

A_{-1}		A_1	
$Stop$	822	$Stop$	48
In^*	-4250	In^*	-981
In	-57	In	-1791
Out	-3053	Out	4010
[to]	5912	[to]	7009
[%␣]	848	[␣the]	3851
[in]	844	[␣be]	2208
[the]	813	[will]	1414
[of]	624	[␣a]	1158
[a]	599	[the]	954

For this reason, the learning process is based on the property $\bullet A_i^x$ which indicates where a link is *not possible*. Since an outbound link on one word is inbound on the other, the inbound/outbound properties of each word are then calculated by a simple bootstrapping process as an average of the opposite properties of the neighboring words.

5 The Weight Function

At each step, the parser must assign a non-negative weight to every candidate link $x \xrightarrow{d} y$ which may be added to an utterance prefix $\langle x_1, \ldots, x_k \rangle$, and the link with the largest (non-zero) weight (with a preference for links between x_{k-1} and x_k) is added to the parse. The weight could be assigned directly based on the In and Out properties of either x or y but this method is not satisfactory for three reasons: first, the values of these properties on low frequency words are not reliable; second, the values of the properties on x and y may conflict; third, some words are ambiguous and require different linking in different contexts. To solve these problems, the weight of the link is taken from the values of In and Out on the best matching label between x and y. This label depends on both words and is usually a frequent word with reliable statistics. It serves as a prototype for the relation between x and y.

5.1 Best Matching Label

A label l is a *matching label* between A_i^x and $A^y_{Sign(-i)}$ if $A_i^x(l) > A_i^x(Stop)$ and either $l = (y, 1)$ or $A^y_{Sign(-i)}(l^{-1}) > 0$. The *best matching label* at A_i^x is the matching label l such that the *match strength* $\min(\bar{A}_i^x(l), \bar{A}^y_{Sign(-i)}(l^{-1}))$ is maximal (if $l = (y, 1)$ then $\bar{A}^y_{Sign(-i)}(l^{-1})$ is defined to be 1). In practice, as before, only the top 10 labels in A_i^x and $A^y_{Sign(-i)}$ are considered.

The *best matching label from x to y* is calculated between A_i^x and $A^y_{Sign(-i)}$ such that A_i^x is on the same side of x as y and was either already used to create a link or is the first adjacency point on that side of x which was not yet used. This means that the adjacency points on each side have to be used one by one, but may be used more than once. The reason is that optional arguments of x usually do not have an adjacency point of their own but have the same labels as obligatory arguments of x and can share their adjacency point. The A_i^x with the strongest matching label is selected, with a preference for the unused adjacency point.

As in the learning process, label matching is blocked between words which are separated by stopping punctuation.

5.2 Calculating the Link Weight

The best matching label $l = (w, \delta)$ from x to y can be either a class ($\delta = 0$) or an adjacency ($\delta = 1$) label at A_i^x. If it is a class label, w can be seen as taking the place of x and all words separating it from y (which are already linked to x). If l is an adjacency label, w can be seen to take the place of y. The calculation of the weight $Wt(x \xrightarrow{d} y)$ of the link from x to y is therefore based on the strengths of the In and Out properties of A_σ^w where $\sigma = Sign(i)$ if $l = (w, 0)$ and $\sigma = Sign(-i)$ if $l = (w, 1)$. In addition, the weight is bounded from above by the best label match strength, $s(l)$:

- If $l = (w, 0)$ and $A_\sigma^w(Out) > 0$:

$$Wt(x \xrightarrow{0} y) = \min(s(l), \bar{A}_\sigma^w(Out))$$

Model	WSJ10			WSJ40			Negra10			Negra40		
	UP	UR	UF_1	UP	UR	UF_1	UP	UR	UF_1	UP	UR	UF_1
Right-branching	55.1	70.0	61.7	35.4	47.4	40.5	33.9	60.1	43.3	17.6	35.0	23.4
Right-branching+punct.	59.1	74.4	65.8	44.5	57.7	50.2	35.4	62.5	45.2	20.9	40.4	27.6
Parsing from POS												
CCM	64.2	81.6	71.9				48.1	85.5	61.6			
DMV+CCM(POS)	69.3	88.0	77.6				49.6	89.7	63.9			
U-DOP	70.8	88.2	78.5			63.9	51.2	90.5	65.4			
UML-DOP			82.9			66.4			67.0			
Parsing from plain text												
DMV+CCM(DISTR.)	65.2	82.8	72.9									
Incremental	75.6	76.2	75.9	58.9	55.9	57.4	51.0	69.8	59.0	34.8	48.9	40.6
Incremental (right to left)	75.9	72.5	74.2	59.3	52.2	55.6	50.4	68.3	58.0	32.9	45.5	38.2

Table 1: Parsing results on WSJ10, WSJ40, Negra10 and Negra40.

- If $l = (w, 1)$:
  - If $A^w_\sigma(In) > 0$:
    $$Wt(x \xrightarrow{d} y) = \min(s(l), \bar{A}^w_\sigma(In))$$
  - Otherwise, if $A^w_\sigma(In^*) \geq |A^w_\sigma(In)|$:
    $$Wt(x \xrightarrow{d} y) = \min(s(l), \bar{A}^w_\sigma(In^*))$$

  where if $A^w_\sigma(In^*) < 0$ and $A^w_\sigma(Out) \leq 0$ then $d = 1$ and otherwise $d = 0$.
- If $A^w_\sigma(Out) \leq 0$ and $A^w_\sigma(In) \leq 0$ and either $l = (w, 1)$ or $A^w_\sigma(Out) = 0$:
  $$Wt(x \xrightarrow{0} y) = s(l)$$
- In all other cases, $Wt(x \xrightarrow{d} y) = 0$.

A link $x \xrightarrow{1} y$ attaches x to y but does not place y inside the smallest bracket covering x. Such links are therefore created in the second case above, when the attachment indication is mixed.

To explain the third case, recall that $s(l) > 0$ means that the label l is stronger than $Stop$ on A^x_i. This implies a link unless the properties of w block it. One way in which w can block the link is to have a positive strength for the link in the opposite direction. Another way in which the properties of w can block the link is if $l = (w, 0)$ and $A^w_\sigma(Out) < 0$, that is, if the learning process has explicitly determined that no outbound link from w (which represents x in this case) is possible. The same conclusion cannot be drawn from a negative value for the In property when $l = (w, 1)$ because, as with standard dependencies, a word determines its outbound links much more strongly than its inbound links.

6 Experiments

The incremental parser was tested on the Wall Street Journal and Negra Corpora.[2] Parsing accuracy was evaluated on the subsets WSJX and Negra$X$ of these corpora containing sentences of length at most X (excluding punctuation). Some of these subsets were used for scoring in (Klein and Manning, 2004; Bod, 2006a; Bod, 2006b). I also use the same precision and recall measures used in those papers: multiple brackets and brackets covering a single word were not counted, but the top bracket was.

The incremental parser learns while parsing, and it could, in principle, simply be evaluated for a single pass of the data. But, because the quality of the parses of the first sentences would be low, I first trained on the full corpus and then measured parsing accuracy on the corpus subset. By training on the full corpus, the procedure differs from that of Klein, Manning and Bod who only train on the subset of bounded length sentences. However, this excludes the induction of parts-of-speech for parsing from plain text. When Klein and Manning induce the parts-of-speech, they do so from a much larger corpus containing the full WSJ treebank together with additional WSJ newswire (Klein and Manning, 2002). The comparison between the algorithms remains, therefore, valid.

Table 1 gives two baselines and the parsing results for WSJ10, WSJ40, Negra10 and Negra40 for recent unsupervised parsing algorithms: CCM

[2] I also tested the incremental parser on the Chinese Treebank version 5.0, achieving an F_1 score of 54.6 on CTB10 and 38.0 on CTB40. Because this version of the treebank is newer and clearly different from that used by previous papers, the results are not comparable and only given here for completeness.

and DMV+CCM (Klein and Manning, 2004), U-DOP (Bod, 2006b) and UML-DOP (Bod, 2006a). The middle part of the table gives results for parsing from part-of-speech sequences extracted from the treebank while the bottom part of the table given results for parsing from plain text. Results for the incremental parser are given for learning and parsing from left to right and from right to left.

The first baseline is the standard right-branching baseline. The second baseline modifies right-branching by using punctuation in the same way as the incremental parser: brackets (except the top one) are not allowed to contain stopping punctuation. It can be seen that punctuation accounts for merely a small part of the incremental parser's improvement over the right-branching heuristic.

Comparing the two algorithms parsing from plain text (of WSJ10), it can be seen that the incremental parser has a somewhat higher combined F_1 score, with better precision but worse recall. This is because Klein and Manning's algorithms (as well as Bod's) always generate binary parse trees, while here no such condition is imposed. The small difference between the recall (76.2) and precision (75.6) of the incremental parser shows that the number of brackets induced by the parser is very close to that of the corpus[3] and that the parser captures the same depth of syntactic structure as that which was used by the corpus annotators.

Incremental parsing from right to left achieves results close to those of parsing from left to right. This shows that the incremental parser has no built-in bias for right branching structures.[4] The slight degradation in performance may suggest that language should not, after all, be processed backwards.

While achieving state of the art accuracy, the algorithm also proved to be fast, parsing (on a 1.86GHz Centrino laptop) at a rate of around 4000 words/sec. and learning (including parsing) at a rate of 3200 – 3600 words/sec. The effect of sentence length on parsing speed is small: the full WSJ corpus was parsed at 3900 words/sec. while WSJ10 was parsed at 4300 words/sec.

[3]The algorithm produced 35588 brackets compared with 35302 brackets in the corpus.

[4]I would like to thank Alexander Clark for suggesting this test.

7 Conclusions

The unsupervised parser I presented here attempts to make use of several universal properties of natural languages: it captures the skewness of syntactic trees in its syntactic representation, restricts the search space by processing utterances incrementally (as humans do) and relies on the Zipfian distribution of words to guide its parsing decisions. It uses an elementary bootstrapping process to deduce the basic properties of the language being parsed. The algorithm seems to successfully capture some of these basic properties, but can be further refined to achieve high quality parsing. The current algorithm is a good starting point for such refinement because it is so very simple.

Acknowledgments I would like to thank Dick de Jongh for many hours of discussion, and Remko Scha, Reut Tsarfaty and Jelle Zuidema for reading and commenting on various versions of this paper.

References

Rens Bod. 2006a. An all-subtrees approach to unsupervised parsing. In *Proceedings of COLING-ACL 2006*.

Rens Bod. 2006b. Unsupervised parsing with U-DOP. In *Proceedings of CoNLL 10*.

Alexander Clark. 2000. Inducing syntactic categories by context distribution clustering. In *Proceedings of CoNLL 4*.

Matthew W. Crocker, Martin Pickering, and Charles Clifton. 2000. *Architectures and Mechanisms for Language Processing*. Cambridge University Press.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of ACL 40*, pages 128–135.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of ACL 42*.

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

Hinrich Schütze. 1995. Distributional part-of-speech tagging. In *Proceedings of EACL 7*.

Patrick Sturt and Matthew W. Crocker. 1996. Monotonic syntactic processing: A cross-linguistic study of attachment and reanalysis. *Language and Cognitive Processes*, 11(5):449–492.

k-best Spanning Tree Parsing

Keith Hall
Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD 21218
keith_hall@jhu.edu

Abstract

This paper introduces a Maximum Entropy dependency parser based on an efficient k-best Maximum Spanning Tree (MST) algorithm. Although recent work suggests that the edge-factored constraints of the MST algorithm significantly inhibit parsing accuracy, we show that generating the 50-best parses according to an edge-factored model has an oracle performance well above the 1-best performance of the best dependency parsers. This motivates our parsing approach, which is based on reranking the k-best parses generated by an edge-factored model. Oracle parse accuracy results are presented for the edge-factored model and 1-best results for the reranker on eight languages (seven from CoNLL-X and English).

1 Introduction

The Maximum Spanning Tree algorithm[1] was recently introduced as a viable solution for non-projective dependency parsing (McDonald et al., 2005b). The dependency parsing problem is naturally a spanning tree problem; however, efficient spanning-tree optimization algorithms assume a cost function which assigns scores independently to edges of the graph. In dependency parsing, this effectively constrains the set of models to those which independently generate parent-child pairs; these are known as edge-factored models. These models are limited to relatively simple features which exclude *linguistic* constructs such as verb sub-categorization/valency, lexical selectional preferences, etc.[2]

In order to explore a rich set of syntactic features in the MST framework, we can either approximate the optimal non-projective solution as in McDonald and Pereira (2006), or we can use the constrained MST model to select a subset of the set of dependency parses to which we then apply less-constrained models. An efficient algorithm for generating the k-best parse trees for a constituency-based parser was presented in Huang and Chiang (2005); a variation of that algorithm was used for generating projective dependency trees for parsing in Dreyer et al. (2006) and for training in McDonald et al. (2005a). However, prior to this paper, an efficient non-projective k-best MST dependency parser has not been proposed.[3]

In this paper we show that the naïve edge-factored models are effective at selecting sets of parses on which the oracle parse accuracy is high. The oracle parse accuracy for a set of parse trees is the highest accuracy for any individual tree in the set. We show that the 1-best accuracy and oracle accuracy can differ by as much as an absolute 9% when the oracle is computed over a small set generated by edge-factored models ($k = 50$).

[1] In this paper we deal only with MSTs on directed graphs. These are often referred to in the graph-theory literature as Maximum Spanning Arborescences.

[2] Labeled edge-factored models can capture selectional preference; however, the unlabeled models presented here are limited to modeling head-child relationships without predicting the *type* of relationship.

[3] The work of McDonald et al. (2005b) would also benefit from a k-best non-projective parser for training.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 392–399,
Prague, Czech Republic, June 2007.

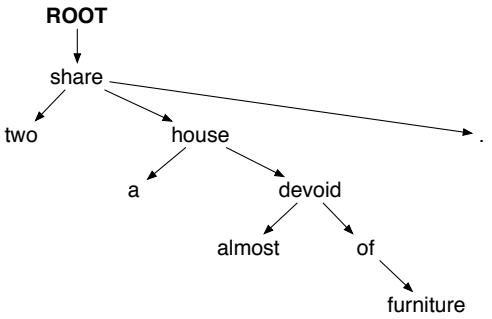


Figure 1: A dependency graph for an English sentence in our development set (Penn WSJ section 24): *Two share a house almost devoid of furniture.*

The combination of two discriminatively trained models, a k-best MST parser and a parse tree reranker, results in an efficient parser that includes complex tree-based features. In the remainder of the paper, we first describe the core of our parser, the k-best MST algorithm. We then introduce the features that we use to compute edge-factored scores as well as tree-based scores. Following, we outline the technical details of our training procedure and finally we present empirical results for the parser on seven languages from the CoNLL-X shared-task and a dependency version of the WSJ Penn Treebank.

2 MST in Dependency Parsing

Work on statistical dependency parsing has utilized either dynamic-programming (DP) algorithms or variants of the Edmonds/Chu-Liu MST algorithm (see Tarjan (1977)). The DP algorithms are generally variants of the CKY bottom-up chart parsing algorithm such as that proposed by Eisner (1996). The Eisner algorithm efficiently ($O(n^3)$) generates projective dependency trees by assembling structures over contiguous words in a clever way to minimize book-keeping. Other DP solutions use constituency-based parsers to produce phrase-structure trees, from which dependency structures are extracted (Collins et al., 1999). A shortcoming of the DP-based approaches is that they are unable to generate non-projective structures. However, non-projectivity is necessary to capture syntactic phenomena in many languages.

McDonald et al. (2005b) introduced a model for dependency parsing based on the Edmonds/Chu-Liu algorithm. The work we present here extends their work by exploring a k-best version of the MST algorithm. In particular, we consider an algorithm proposed by Camerini et al. (1980) which has a worst-case complexity of $O(km\log(n))$, where k is the number of parses we want, n is the number of words in the input sentence, and m is the number of edges in the hypothesis graph. This can be reduced to $O(kn^2)$ in dense graphs[4] by choosing appropriate data structures (Tarjan, 1977). Under the models considered here, all pairs of words are considered as candidate parents (children) of another, resulting in a fully connected graph, thus $m = n^2$.

In order to incorporate second-order features (specifically, sibling features), McDonald et al. proposed a dependency parser based on the Eisner algorithm (McDonald and Pereira, 2006). The second-order features allow for more complex phrasal relationships than the edge-factored features which only include parent/child features. Their algorithm finds the best solution according to the Eisner algorithm and then searches for the single valid edge change that increases the tree score. The algorithm iterates until no better single edge substitution can improve the score of the tree. This greedy approximation allows for second-order constraints and non-projectivity. They found that applying this method to trees generated by the Eisner algorithm using second-order features performs better than applying it to the best tree produced by the MST algorithm with first-order (edge-factored) features.

In this paper we provide a new evaluation of the efficacy of edge-factored models, k-best oracle results. We show that even when k is small, the edge-factored models select k-best sets which contain good parses. Furthermore, these good parses are even better than the parses selected by the best dependency parsers.

2.1 k-best MST Algorithm

The k-best MST algorithm we introduce in this paper is the algorithm described in Camerini et al. (1980). For proofs of complexity and correctness, we defer to the original paper. This section is intended to provide the intuitions behind the algorithm and allow for an understanding of the key data-structures necessary to ensure the theoretical guarantees.

[4] A dense graph is one in which the number of edges is close to the number of edges in a fully connected graph (i.e., n^2).

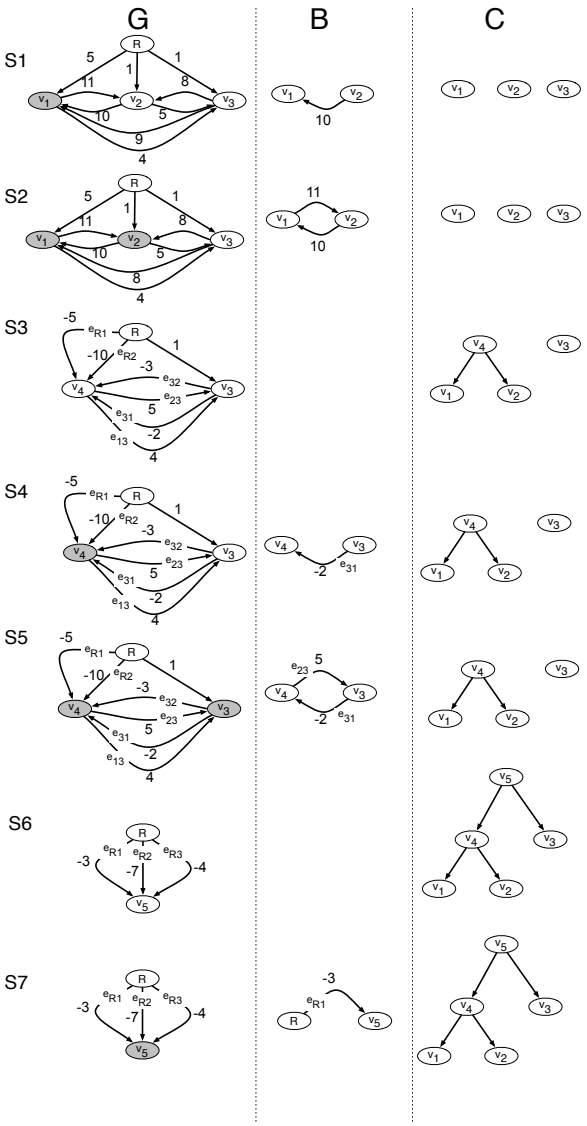


Figure 2: Simulated 1-best MST algorithm.

Let $G = \{V, E\}$ be a directed graph where $V = \{R, v_1, \ldots, v_n\}$ and $E = \{e_{11}, e_{12}, \ldots, e_{1n}, e_{21}, \ldots, e_{nn}\}$. We refer to edge e_{ij} as the edge that is directed from v_i into v_j in the graph. The initial dependency graph in Figure 2 (column **G**) contains three regular nodes and a root node.

Algorithm 1 is a version of the MST algorithm as presented by Camerini et al. (1980); subtleties of the algorithm have been omitted. Arguments Y (a branching[5]) and Z (a set of edges) are constraints on the edges that can be part of the solution, A. Edges in Y are required to be in the solution and edges in

[5]A branching is a subgraph that contains no cycles and no more than one edge directed into each node.

Algorithm 1 Sketch of 1-best MST algorithm

```
procedure BEST(G, Y, Z)
    G = (G ∪ Y) − Z
    B = ∅
    C = V
 5: for unvisited vertex v_i ∈ V do
        mark v_i as visited
        get best in-edge b ∈ {e_jk : k = i} for v_i
        B = B ∪ b
        β(v_i) = b
10:     if B contains a cycle C then
            create a new node v_{n+1}
            C = C ∪ v_{n+1}
            make all nodes of C children of v_{n+1} in C
            COLLAPSE all nodes of C into v_{n+1}
15:         ADD v_{n+1} to list of unvisited vertices
            n = n + 1
            B = B − C
        end if
    end for
20: EXPAND C choosing best way to break cycles
    Return best A = {b ∈ E|∃v ∈ V : β(v) = b}
      and C
end procedure
```

Z cannot be part of the solution. The branching C stores a hierarchical history of cycle collapses, encapsulating embedded cycles and allowing for an *expanding* procedure, which breaks cycles while maintaining an optimal solution.

Figure 2 presents a view of the algorithm when run on a three node graphs (plus a specified root node). Steps S1, S2, S4, and S5 depict the processing of lines 5 to 8, recording in β the best input edges for each vertex. Steps S3 and S6 show the process of *collapsing* a cycle into a new node (lines 10 to 16).

The main loop of the algorithm processes each vertex that has not yet been visited. We look up the best incoming edge (which is stored in a priority-queue). This value is recorded in β and the edge is added to the current *best* graph B. We then check to see if adding this new edge would create a cycle in B. If so, we create a new node and *collapse* the cycle into it. This can be seen in Step S3 in Figure 2.

The process of collapsing a cycle into a node involves removing the edges in the cycle from B, and adjusting the weights of all edges directed into any node in the cycle. The weights are adjusted so that they reflect the relative difference of choosing the new in-edge rather than the edge in the cycle. In step S3, observe that edge e_{R1} had a weight of 5, but now that it points into the new node v_4, we subtract the weight of the edge e_{21} that also pointed into v_1,

which was 10. Additionally, we record in C the relationship between the new node v_4 and the original nodes v_1 and v_2.

This process continues until we have visited all original and newly created nodes. At that point, we *expand* the cycles encoded in C. For each node not originally in G (e.g., v_5, v_4), we retrieve the edge e_r pointing into this node, recorded in β. We identify the node v_s to which e_r pointed in the original graph G and set $\beta(v_s) = e_r$.

Algorithm 2 Sketch of next-best MST algorithm

```
    procedure NEXT(G, Y, Z, A, C)
        δ ← +∞
        for unvisited vertex v do
            get best in-edge b for v
 5:         if b ∈ A − Y then
                f ← alternate edge into v
                if swapping f with b results in smaller δ then
                    update δ, let e ← f
                end if
10:         end if
            if b forms a cycle then
                Resolve as in 1-best
            end if
        end for
15:     Return edge e and δ
    end procedure
```

Algorithm 2 returns the single edge, e, of the 1-best solution A that, when removed from the graph, results in a graph for which the best solution is the next best solution after A. Additionally, it returns δ, the difference in score between A and the next best tree. The branching C is passed in from Algorithm 1 and is used here to efficiently identify alternate edges, f, for edge e.

Y and Z in Algorithms 1 and 2 are used to construct the next best solutions efficiently. We call $G_{Y,Z}$ a constrained graph; the constraints being that Y restricts the in-edges for a subset of nodes: for each vertex with an in-edge in Y, only the edge of Y can be an in-edge of the vertex. Also, edges in Z are removed from the graph. A constrained spanning tree for $G_{Y,Z}$ (a tree covering all nodes in the graph) must satisfy: $Y \subseteq A \subseteq E - Z$.

Let A be the (constrained) solution to a (constrained) graph and let e be the edge that leads to the next best solution. The third-best solution is either the second-best solution to $G_{Y,\{Z \cup e\}}$ or the second-best solution to $G_{\{Y \cup e\},Z}$. The k-best ranking algorithm uses this fact to incrementally partition the solution space: for each solution, the next best either will include e or will not include e.

Algorithm 3 k-best MST ranking algorithm

```
    procedure RANK(G, k)
        A, C ← best(E, V, ∅, ∅)
        (e, δ) ← next(E, V, ∅, ∅, A, C)
        bestList ← A
 5:     Q ← enqueue(s(A) − δ, e, A, C, ∅, ∅)
        for j ← 2 to k do
            (s, e, A, C, Y, Z) = dequeue(Q)
            Y′ = Y ∪ e
            Z′ = Z ∪ e
10:         A′, C′ ← best(E, V, Y, Z′)
            bestList ← A′
            e′, δ′ ← next(E, V, Y′, Z, A′, C′)
            Q ← enqueue(s(A) − δ′, e′, A′, C′, Y′, Z)
            e′, δ′ ← next(E, V, Y, Z′, A′, C′)
15:         Q ← enqueue(s(A) − δ′, e′, A′, C′, Y, Z′)
        end for
        Return bestList
    end procedure
```

The k-best ranking procedure described in Algorithm 3 uses a priority queue, Q, keyed on the first parameter to $enqueue$ to keep track of the horizon of next best solutions. The function $s(A)$ returns the score associated with the tree A. Note that in each iteration there are two new elements enqueued representing the sets $G_{Y,\{Z \cup e\}}$ and $G_{\{Y \cup e\},Z}$.

Both Algorithms 1 and 2 run in $O(m \log(n))$ time and can run in quadratic time for dense graphs with the use of an efficient priority-queue[6] (i.e., based on a Fibonacci heap). Algorithm 3 runs in constant time, resulting in an $O(km \log n)$ algorithm (or $O(kn^2)$ for dense graphs).

3 Dependency Models

Each of the two stages of our parser is based on a discriminative training procedure. The edge-factored model is based on a conditional log-linear model trained using the Maximum Entropy constraints.

3.1 Edge-factored MST Model

One way in which dependency parsing differs from constituency parsing is that there is a fixed amount of structure in every tree. A dependency tree for a sentence of n words has exactly n edges,[7] each repre-

[6] Each vertex keeps a priority queue of candidate parents. When a cycles is collapsed, the new vertex inherits the union of queues associated with the vertices of the cycle.

[7] We assume each tree has a *root* node.

senting a *syntactic* or *semantic* relationship, depending on the linguistic model assumed for annotation. A spanning tree (equivalently, a dependency parse) is a subgraph for which each node has one in-edge, the root node has zero in-edges, and there are no cycles.

Edge-factored features are defined over the edge and the input sentence. For each of the n^2 parent/child pairs, we extract the following features:

Node-type There are three basic node-type features: word *form*, morphologically reduced *lemma*, and part-of-speech (*POS*) tag. The CoNLL-X data format[8] describes two part-of-speech tag types, we found that features derived from the coarse tags are more reliable. We consider both unigram (parent or child) and bigram (composite parent/child) features. We refer to parent features with the prefix p- and child feature with the prefix c-; for example: p–pos, p–form, c–pos, and c–form. In our model we use both word form and POS tag and include the composite form/POS features: p–form/c–pos and p–pos/c–form.

Branch A binary feature which indicates whether the child is to the left or right of the parent in the input string. Additionally, we provide composite features p–pos/branch and p–pos/c–pos/branch.

Distance The number of words occurring between the parent and child word. These distances are bucketed into 7 buckets (1 through 6 plus an additional single bucket for distances greater than 6). Additionally, this feature is combined with node-type features: p–pos/dist, c–pos/dist, p–pos/c–pos/dist.

Inside POS tags of the words between the parent and child. A count of each tag that occurs is recorded, the feature is identified by the tag and the feature value is defined by the count. Additional composite features are included combining the inside and node-type: for each type t_i the composite features are: p–pos/t_i, c–pos/t_i, p–pos/c–pos/t_i.

Outside Exactly the same as the **Inside** feature except that it is defined over the features to the left and right of the span covered by this parent-child pair.

Extra-Feats Attribute-value pairs from the CoNLL *FEATS* field including combinations with parent/child node-types. These features represent word-level annotations provided in the treebank and include morphological and lexical-semantic features. These do not exist in the English data.

Inside Edge Similar to *Inside* features, but only includes nodes immediately to left and right within the span covered by the parent/child pair. We include the following features where i^l and i^r are the inside left and right POS tags and i^p is the inside POS tag closest to the parent: i^l/i^r, p–pos/i^p, p–pos/i^l/i^r/c–pos,

Outside Edge An *Outside* version of the *Inside Edge* feature type.

Many of the features above were introduced in McDonald et al. (2005a); specifically, the *node-type*, *inside*, and *edge* features. The number of features can grow quite large when form or lemma features are included. In order to handle large training sets with a large number of features we introduce a bagging-based approach, described in Section 4.2.

3.2 Tree-based Reranking Model

The second stage of our dependency parser is a reranker that operates on the output of the k-best MST parser. Features in this model are not constrained as in the edge-factored model. Many of the model features have been inspired by the constituency-based features presented in Charniak and Johnson (2005). We have also included features that exploit non-projectivity where possible. The node-type is the same as defined for the MST model.

MST score The score of this parse given by the first-stage MST model.

Sibling The POS-tag of immediate siblings. Intended to capture the preference for particular immediate siblings such as modifiers.

Valency Count of the number of children for each word (indexed by POS-tag of the word). These

[8] The 2006 CoNLL-X data format can be found on-line at: `http://nextens.uvt.nl/~conll/`.

counts are bucketed into 4 buckets. For example, a feature may look like p–pos=VB/v=4, meaning the POS tag of the parent is 'VB' and it had 4 dependents.

Sub-categorization A string representing the sequence of child POS tags for each parent POS-tag.

Ancestor Grandparent and great grandparent POS-tag for each word. Composite features are generated with the label c–pos/p–pos/gp–pos and c–pos/p–pos/ggp–pos (where gp is the grand-parent and ggp is the great grand-parent).

Edge POS-tag to the left and right of the subtree, both inside and outside the subtree. For example, say a subtree with parent POS-tag p–pos spans from i to j, we include composite *outside* features: p–pos/n_{i-1}–pos/n_{j+1}–pos, p–pos/n_{i-1}–pos, p–pos/n_{j+1}–pos; and composite inside features: p–pos/n_{i+1}–pos/n_{j-1}–pos, p–pos/n_{i+1}–pos, p–pos/n_{j-1}–pos.

Branching Factor Average number of left/right branching nodes per POS-tag. Additionally, we include a boolean feature indicating the overall left/right preference.

Depth Depth of the tree and depth normalized by sentence length.

Heavy Number of dominated nodes per POS-tag. We also include the average number of nodes dominated by each POS-tag.

4 MaxEnt Training

We have adopted the conditional Maximum Entropy (MaxEnt) modeling paradigm as outlined in Charniak and Johnson (2005) and Riezler et al. (2002). We can partition the training examples into independent subsets, Y_s: for the edge-factored MST models, each set represents a word and its candidate parents; for the reranker, each set represents the k-best trees for a particular sentence. We wish to estimate the conditional distribution over hypotheses in the set y_i, given the set: $p(y_i|Y_s) = \frac{\exp(\sum_k \lambda_k f_{ik})}{\sum_{j:y_j \in Y_s} \exp(\sum_{k'} \lambda k' f_{jk'})}$, where f_{ik} is the k^{th} feature function in the model for example y_i.

4.1 MST Training

Our MST parser training procedure involves enumerating the n^2 potential tree edges (parent/child pairs). Unlike the training procedure employed by McDonald et al. (2005b) and McDonald and Pereira (2006), we provide positive and negative examples in the training data. A node can have at most one parent, providing a natural split of the n^2 training examples. For each node n_i, we wish to estimate a distribution over n nodes[9] as potential parents, $p(v_i, e_{ji}|e_{\_i})$, the probability of the correct parent of v_i being v_j given the set of edges associated with its candidate parents $e_{\_i}$. We call this the parent-prediction model.

4.2 MST Bagging

The complexity of the training procedure is a function of the number of features and the number of examples. For large datasets, we use an ensemble technique inspired by *Bagging* (Breiman, 1996). Bagging is generally used to mitigate high variance in datasets by sampling, with replacement, from the training set. Given that we wish to include some of the less frequent examples and therefore are not necessarily avoiding high variance, we partition the data into disjoint sets.

For each of the sets, we train a model independently. Furthermore, we only allow the parameters to be changed for those features observed in the training set. At inference time, we apply each model to the training data and then combine the prediction probabilities.

$$\tilde{p}_\theta(y_i|Y_s) = \max_m p_{\theta_m}(y_i|Y_s) \quad (1)$$

$$\tilde{p}_\theta(y_i|Y_s) = \frac{1}{M}\sum_m p_{\theta_m}(y_i|Y_s) \quad (2)$$

$$\tilde{p}_\theta(y_i|Y_s) = \left(\prod_m p_{\theta_m}(y_i|Y_s)\right)^{1/M} \quad (3)$$

$$\tilde{p}_\theta(y_i|Y_s) = \frac{M}{\sum_m \frac{1}{p_{\theta_m}(y_i|Y_s)}} \quad (4)$$

Equations 1, 2, 3, and 4 are the maximum, average, geometric mean, and harmonic mean, respectively. We performed an exploration of these on the

[9] Recall that in addition to the $n-1$ other nodes in the graph, there is a root node for which we know has no parents.

development data and found that the geometric mean produces the best results (Equation 3); however, we observed only very small differences in the accuracy among models where only the combination function differed.

4.3 Reranker Training

The second stage of parsing is performed by our tree-based reranker. The input to the reranker is a list of k parses generated by the k-best MST parser. For each input sentence, the hypothesis set is the k parses. At inference time, predictions are made independently for each hypothesis set Y_s and therefore the normalization factor can be ignored.

5 Empirical Evaluation

The CoNLL-X shared task on dependency parsing provided data for a number of languages in a common data format. We have selected seven of these languages for which the data is available to us. Additionally, we have automatically generated a dependency version of the Penn WSJ treebank.[10] As we are only interested in the structural component of a parse in this paper, we present results for unlabeled dependency parsing. A second labeling stage can be applied to get labeled dependency structures as described in (McDonald et al., 2006).

In Table 1 we report the accuracy for seven of the CoNLL languages and English.[11] Already, at $k = 50$, we see the oracle rate climb as much as 9.25% over the 1-best result (Dutch). Continuing to increase the size of the k-best lists adds to the oracle accuracy, but the relative improvement appears to be increasing at a logarithmic rate. The k-best parser is used both to train the k-best reranker and, at inference time, to select a set of hypotheses to rerank. It is not necessary that training is done with the same size hypothesis set as test, we explore the matched and mismatched conditions in our reranking experiments.

[10]The Penn WSJ treebank was converted using the conversion program described in (Johansson and Nugues, 2007) and available on the web at: `http://nlp.cs.lth.se/pennconverter/`

[11]The **Best Reported** results is from the CoNLL-X competition. The best result reported for English is the Charniak parser (without reranking) on Section 23 of the WSJ Treebank using the same head-finding rules as for the evaluation data.

Table 2 shows the reranking results for the set of languages. For each language, we select model parameters on a development set prior to running on the test data. These parameters include a feature count threshold (the minimum number of observations of a feature before it is included in a model) and a mixture weight controlling the contribution of a quadratic regularizer (used in MaxEnt training). For Czech, German, and English, we use the MST bagging technique with 10 bags. These test results are for the models which performed best on the development set (using 50-best parses).

We see minor improvements over the 1-best baseline MST output (repeated in this table for comparison). We believe this is due to the overwhelming number of parameters in the reranking models and the relatively small amount of training data. Interestingly, increasing the number of hypotheses helps for some languages and hurts the others.

6 Conclusion

Although the edge-factored constraints of MST parsers inhibit accuracy in 1-best parsing, edge-factored models are effective at selecting high accuracy k-best sets. We have introduced the Camerini et al. (1980) k-best MST algorithm and have shown how to efficiently train MaxEnt models for dependency parsing. Additionally, we presented a unified modeling and training setting for our two-stage parser; MaxEnt training is used to estimate the parameters in both models. We have introduced a particular ensemble technique to accommodate the large training sets generated by the first-stage edge-factored modeling paradigm. Finally, we have presented a reranker which attempts to select the best tree from the k-best set. In future work we wish to explore more robust feature sets and experiment with feature selection techniques to accommodate them.

Acknowledgments

This work was partially supported by U.S. NSF grants IIS–9982329 and OISE–0530118. We thank Ryan McDonald for directing us to the Camerini et al. paper and Liang Huang for insightful comments.

Language	Best	Oracle Accuracy				
	Reported	$k = 1$	$k = 10$	$k = 50$	$k = 100$	$k = 500$
Arabic	79.34	77.92	80.72	82.18	83.03	84.47
Czech	87.30	83.56	88.50	90.88	91.80	93.50
Danish	90.58	89.12	92.89	94.79	95.29	96.59
Dutch	83.57	81.05	87.43	90.30	91.28	93.12
English	92.36	85.04	89.04	91.12	91.87	93.42
German	90.38	87.02	91.51	93.39	94.07	95.47
Portuguese	91.36	89.86	93.11	94.85	95.39	96.47
Swedish	89.54	86.50	91.20	93.37	93.83	95.42

Table 1: k-best MST oracle results. The 1-best results represent the performance of the parser in isolation. Results are reported for the CoNLL test set and for English, on Section 23 of the Penn WSJ Treebank.

Language	Best	Reranked Accuracy				
	Reported	1-best	10-best	50-best	100-best	500-best
Arabic	79.34	77.61	78.06	78.02	77.94	77.76
Czech	87.30	83.56	83.94	84.14	84.48	84.46
Danish	90.58	89.12	89.48	89.76	89.68	89.74
Dutch	83.57	81.05	82.01	82.91	82.83	83.21
English	92.36	85.04	86.54	87.22	87.38	87.81
German	90.38	87.02	88.24	88.72	88.76	88.90
Portuguese	91.36	89.38	90.00	89.98	90.02	90.02
Swedish	89.54	86.50	87.87	88.21	88.26	88.53

Table 2: Second-stage results from the k-best parser and reranker. The **Best Reported** and 1-best fields are copied from table 1. Only non-lexical features were used for the reranking models.

References

Leo Breiman. 1996. Bagging predictors. *Machine Learning*, 26(2):123–140.

Paolo M. Camerini, Luigi Fratta, and Francesco Maffioli. 1980. The k best spanning arborescences of a network. *Networks*, 10:91–110.

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

Michael Collins, Lance Ramshaw, Jan Hajič, and Christoph Tillmann. 1999. A statistical parser for Czech. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, pages 505–512.

Markus Dreyer, David A. Smith, and Noah A. Smith. 2006. Vine parsing and minimum risk reranking for speed and precision. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages 340–345.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the 9th International Workshop on Parsing Technologies*.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of NODALIDA 2007*, Tartu, Estonia, May 25-26. To appear.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the Annual Meeting of the European Association for Computational Linguistics*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics*.

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

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency parsing with a two-stage discriminative parser. In *Conference on Natural Language Learning*.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. III Maxwell, and Mark Johnson. 2002. Parsing the Wall Street Journal using a lexical-functional grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Morgan Kaufmann.

R.E. Tarjan. 1977. Finding optimal branchings. *Networks*, 7:25–35.

Is the End of Supervised Parsing in Sight?

Rens Bod
School of Computer Science
University of St Andrews, ILLC, University of Amsterdam
rb@cs.st-and.ac.uk

Abstract

How far can we get with unsupervised parsing if we make our training corpus several orders of magnitude larger than has hitherto be attempted? We present a new algorithm for unsupervised parsing using an all-subtrees model, termed U-DOP*, which parses directly with packed forests of all binary trees. We train both on Penn's WSJ data and on the (much larger) NANC corpus, showing that U-DOP* outperforms a treebank-PCFG on the standard WSJ test set. While U-DOP* performs worse than state-of-the-art supervised parsers on hand-annotated sentences, we show that the model outperforms supervised parsers when evaluated as a language model in syntax-based machine translation on Europarl. We argue that supervised parsers miss the fluidity between constituents and non-constituents and that in the field of syntax-based language modeling the end of supervised parsing has come in sight.

1 Introduction

A major challenge in natural language parsing is the unsupervised induction of syntactic structure. While most parsing methods are currently supervised or semi-supervised (McClosky et al. 2006; Henderson 2004; Steedman et al. 2003), they depend on hand-annotated data which are difficult to come by and which exist only for a few languages. Unsupervised parsing methods are becoming increasingly important since they operate with raw, unlabeled data of which unlimited quantities are available.

There has been a resurgence of interest in unsupervised parsing during the last few years. Where van Zaanen (2000) and Clark (2001) induced unlabeled phrase structure for small domains like the ATIS, obtaining around 40% unlabeled f-score, Klein and Manning (2002) report 71.1% f-score on Penn WSJ part-of-speech strings ≤ 10 words (WSJ10) using a constituent-context model called CCM. Klein and Manning (2004) further show that a hybrid approach which combines constituency and dependency models, yields 77.6% f-score on WSJ10.

While Klein and Manning's approach may be described as an "all-substrings" approach to unsupervised parsing, an even richer model consists of an "all-subtrees" approach to unsupervised parsing, called U-DOP (Bod 2006). U-DOP initially assigns all unlabeled binary trees to a training set, efficiently stored in a packed forest, and next trains subtrees thereof on a held-out corpus, either by taking their relative frequencies, or by iteratively training the subtree parameters using the EM algorithm (referred to as "UML-DOP"). The main advantage of an all-subtrees approach seems to be the direct inclusion of *discontiguous* context that is not captured by (linear) substrings. Discontiguous context is important not only for learning structural dependencies but also for learning a variety of non-contiguous constructions such as *nearest … to…* or *take … by surprise*. Bod (2006) reports 82.9% unlabeled f-score on the same WSJ10 as used by Klein and Manning (2002, 2004). Unfortunately, his experiments heavily depend on a priori sampling of subtrees, and the model becomes highly inefficient if larger corpora are used or longer sentences are included.

In this paper we will also test an alternative model for unsupervised all-subtrees

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 400–407,
Prague, Czech Republic, June 2007.

parsing, termed U-DOP*, which is based on the DOP* estimator by Zollmann and Sima'an (2005), and which computes the shortest derivations for sentences from a held-out corpus using all subtrees from all trees from an extraction corpus. While we do not achieve as high an f-score as the UML-DOP model in Bod (2006), we will show that U-DOP* can operate without subtree sampling, and that the model can be trained on corpora that are two orders of magnitude larger than in Bod (2006). We will extend our experiments to 4 million sentences from the NANC corpus (Graff 1995), showing that an f-score of 70.7% can be obtained on the standard Penn WSJ test set by means of unsupervised parsing. Moreover, U-DOP* can be directly put to use in bootstrapping structures for concrete applications such as syntax-based machine translation and speech recognition. We show that U-DOP* outperforms the supervised DOP model if tested on the German-English Europarl corpus in a syntax-based MT system.

In the following, we first explain the DOP* estimator and discuss how it can be extended to unsupervised parsing. In section 3, we discuss how a PCFG reduction for supervised DOP can be applied to packed parse forests. In section 4, we will go into an experimental evaluation of U-DOP* on annotated corpora, while in section 5 we will evaluate U-DOP* on unlabeled corpora in an MT application.

2 From DOP* to U-DOP*

DOP* is a modification of the DOP model in Bod (1998) that results in a statistically consistent estimator and in an efficient training procedure (Zollmann and Sima'an 2005). DOP* uses the all-subtrees idea from DOP: given a treebank, take all subtrees, regardless of size, to form a stochastic tree-substitution grammar (STSG). Since a parse tree of a sentence may be generated by several (leftmost) derivations, the probability of a tree is the sum of the probabilities of the derivations producing that tree. The probability of a derivation is the product of the subtree probabilities. The original DOP model in Bod (1998) takes the occurrence frequencies of the subtrees in the trees normalized by their root frequencies as subtree parameters. While efficient algorithms have been developed for this DOP model by converting it into a PCFG reduction (Goodman 2003), DOP's estimator was shown to be inconsistent by Johnson (2002). That is, even with unlimited training data, DOP's estimator is not guaranteed to converge to the correct distribution.

Zollmann and Sima'an (2005) developed a statistically consistent estimator for DOP which is based on the assumption that maximizing the joint probability of the parses in a treebank can be approximated by maximizing the joint probability of their shortest derivations (i.e. the derivations consisting of the fewest subtrees). This assumption is in consonance with the principle of simplicity, but there are also empirical reasons for the shortest derivation assumption: in Bod (2003) and Hearne and Way (2006), it is shown that DOP models that select the preferred parse of a test sentence using the shortest derivation criterion perform very well.

On the basis of this shortest-derivation assumption, Zollmann and Sima'an come up with a model that uses held-out estimation: the training corpus is randomly split into two parts proportional to a fixed ratio: an *extraction corpus* EC and a *held-out* corpus HC. Applied to DOP, held-out estimation would mean to extract fragments from the trees in EC and to assign their weights such that the likelihood of HC is maximized. If we combine their estimation method with Goodman's reduction of DOP, Zollman and Sima'an's procedure operates as follows:

(1) Divide a treebank into an EC and HC
(2) Convert the subtrees from EC into a PCFG reduction
(3) Compute the shortest derivations for the sentences in HC (by simply assigning each subtree equal weight and applying Viterbi 1-best)
(4) From those shortest derivations, extract the subtrees and their relative frequencies in HC to form an STSG

Zollmann and Sima'an show that the resulting estimator is consistent. But equally important is the fact that this new DOP* model does not suffer from a decrease in parse accuracy if larger subtrees are included, whereas the original DOP model needs to be redressed by a correction factor to maintain this property (Bod 2003). Moreover, DOP*'s estimation procedure is very efficient, while the EM training procedure for UML-DOP

proposed in Bod (2006) is particularly time consuming and can only operate by randomly sampling trees.

Given the advantages of DOP*, we will generalize this model in the current paper to *unsupervised* parsing. We will use the same all-subtrees methodology as in Bod (2006), but now by applying the efficient and consistent DOP*-based estimator. The resulting model, which we will call U-DOP*, roughly operates as follows:

(1) Divide a corpus into an EC and HC
(2) Assign all unlabeled binary trees to the sentences in EC, and store them in a shared parse forest
(3) Convert the subtrees from the parse forests into a compact PCFG reduction (see next section)
(4) Compute the shortest derivations for the sentences in HC (as in DOP*)
(5) From those shortest derivations, extract the subtrees and their relative frequencies in HC to form an STSG
(6) Use the STSG to compute the most probable parse trees for new test data by means of Viterbi *n*-best (see next section)

We will use this U-DOP* model to investigate our main research question: *how far can we get with unsupervised parsing if we make our training corpus several orders of magnitude larger than has hitherto be attempted?*

3 Converting shared parse forests into PCFG reductions

The main computational problem is how to deal with the immense number of subtrees in U-DOP*. There exists already an efficient *supervised* algorithm that parses a sentence by means of all subtrees from a treebank. This algorithm was extensively described in Goodman (2003) and converts a DOP-based STSG into a compact PCFG reduction that generates eight rules for each node in the treebank. The reduction is based on the following idea: every node in every treebank tree is assigned a unique number which is called its address. The notation $A@k$ denotes the node at address k where A is the nonterminal labeling that node. A new nonterminal is created for each node in the training data. This nonterminal is called A_k. Let a_j represent the number of subtrees headed by the node $A@j$, and let a represent the number of subtrees headed by nodes with nonterminal A, that is $a = \Sigma_j\ a_j$. Then there is a PCFG with the following property: for every subtree in the training corpus headed by A, the grammar will generate an isomorphic subderivation. For example, for a node $(A@j\ (B@k,\ C@l))$, the following eight PCFG rules in figure 1 are generated, where the number following a rule is its weight.

$A_j \rightarrow BC$	$(1/a_j)$	$A \rightarrow BC$	$(1/a)$
$A_j \rightarrow B_kC$	(b_k/a_j)	$A \rightarrow B_kC$	(b_k/a)
$A_j \rightarrow BC_l$	(c_l/a_j)	$A \rightarrow BC_l$	(c_l/a)
$A_j \rightarrow B_kC_l$	(b_kc_l/a_j)	$A \rightarrow B_kC_l$	(b_kc_l/a)

Figure 1. PCFG reduction of supervised DOP

By simple induction it can be shown that this construction produces PCFG derivations isomorphic to DOP derivations (Goodman 2003: 130-133). The PCFG reduction is linear in the number of nodes in the corpus.

While Goodman's reduction method was developed for supervised DOP where each training sentence is annotated with exactly one tree, the method can be generalized to a corpus where each sentence is annotated with all possible binary trees (labeled with the generalized category X), as long as we represent these trees by a shared parse forest. A shared parse forest can be obtained by adding pointers from each node in the chart (or tabular diagram) to the nodes that caused it to be placed in the chart. Such a forest can be represented in cubic space and time (see Billot and Lang 1989). Then, instead of assigning a unique address to each node in each tree, as done by the PCFG reduction for supervised DOP, we now assign a unique address to each node in each parse forest for each sentence. However, the same node may be part of more than one tree. A shared parse forest is an AND-OR graph where AND-nodes correspond to the usual parse tree nodes, while OR-nodes correspond to distinct subtrees occurring in the same context. The total number of nodes is cubic in sentence length n. This means that there are $O(n^3)$ many nodes that receive a unique address as described above, to which next our PCFG reduction is applied. This is a huge reduction compared to Bod (2006) where

the number of subtrees of all trees increases with the Catalan number, and only ad hoc sampling could make the method work.

Since U-DOP* computes the shortest derivations (in the training phase) by combining subtrees from unlabeled binary trees, the PCFG reduction in figure 1 can be represented as in figure 2, where *X* refers to the generalized category while *B* and *C* either refer to part-of-speech categories or are equivalent to *X*. The equal weights follow from the fact that the shortest derivation is equivalent to the most probable derivation if all subtrees are assigned equal probability (see Bod 2000; Goodman 2003).

$X_j \rightarrow BC$	1	$X \rightarrow BC$	0.5
$X_j \rightarrow B_kC$	1	$X \rightarrow B_kC$	0.5
$X_j \rightarrow BC_l$	1	$X \rightarrow BC_l$	0.5
$X_j \rightarrow B_kC_l$	1	$X \rightarrow B_kC_l$	0.5

Figure 2. PCFG reduction for U-DOP*

Once we have parsed HC with the shortest derivations by the PCFG reduction in figure 2, we extract the subtrees from HC to form an STSG. The number of subtrees in the shortest derivations is linear in the number of nodes (see Zollmann and Sima'an 2005, theorem 5.2). This means that U-DOP* results in an STSG which is much more succinct than previous DOP-based STSGs. Moreover, as in Bod (1998, 2000), we use an extension of Good-Turing to smooth the subtrees and to deal with 'unknown' subtrees.

Note that the direct conversion of parse forests into a PCFG reduction also allows us to efficiently implement the maximum likelihood extension of U-DOP known as UML-DOP (Bod 2006). This can be accomplished by training the PCFG reduction on the held-out corpus HC by means of the expectation-maximization algorithm, where the weights in figure 1 are taken as initial parameters. Both U-DOP*'s and UML-DOP's estimators are known to be statistically consistent. But while U-DOP*'s training phase merely consists of the computation of the shortest derivations and the extraction of subtrees, UML-DOP involves iterative training of the parameters.

Once we have extracted the STSG, we compute the most probable parse for new sentences by Viterbi *n*-best, summing up the probabilities of derivations resulting in the same tree (the exact computation of the most probable parse is NP hard – see Sima'an 1996). We have incorporated the technique by Huang and Chiang (2005) into our implementation which allows for efficient Viterbi *n*-best parsing.

4 Evaluation on hand-annotated corpora

To evaluate U-DOP* against UML-DOP and other unsupervised parsing models, we started out with three corpora that are also used in Klein and Manning (2002, 2004) and Bod (2006): Penn's WSJ10 which contains 7422 sentences ≤ 10 words after removing empty elements and punctuation, the German NEGRA10 corpus and the Chinese Treebank CTB10 both containing 2200+ sentences ≤ 10 words after removing punctuation. As with most other unsupervised parsing models, we train and test on p-o-s strings rather than on word strings. The extension to word strings is straightforward as there exist highly accurate unsupervised part-of-speech taggers (e.g. Schütze 1995) which can be directly combined with unsupervised parsers, but for the moment we will stick to p-o-s strings (we will come back to word strings in section 5). Each corpus was divided into 10 training/test set splits of 90%/10% (*n*-fold testing), and each training set was randomly divided into two equal parts, that serve as EC and HC and vice versa. We used the same evaluation metrics for unlabeled precision (UP) and unlabeled recall (UR) as in Klein and Manning (2002, 2004). The two metrics of UP and UR are combined by the unlabeled f-score F1 = 2*UP*UR/(UP+UR). All trees in the test set were binarized beforehand, in the same way as in Bod (2006).

For UML-DOP the decrease in cross-entropy became negligible after maximally 18 iterations. The training for U-DOP* consisted in the computation of the shortest derivations for the HC from which the subtrees and their relative frequencies were extracted. We used the technique in Bod (1998, 2000) to include 'unknown' subtrees. Table 1 shows the f-scores for U-DOP* and UML-DOP against the f-scores for U-DOP reported in Bod (2006), the CCM model in Klein and Manning (2002), the DMV dependency model in Klein and Manning (2004) and their combined model DMV+CCM.

Model	English (WSJ10)	German (NEGRA10)	Chinese (CTB10)
CCM	71.9	61.6	45.0
DMV	52.1	49.5	46.7
DMV+CCM	77.6	63.9	43.3
U-DOP	78.5	65.4	46.6
U-DOP*	77.9	63.8	42.8
UML-DOP	79.4	65.2	45.0

Table 1. F-scores of U-DOP* and UML-DOP compared to other models on the same data.

It should be kept in mind that an exact comparison can only be made between U-DOP* and UML-DOP in table 1, since these two models were tested on 90%/10% splits, while the other models were applied to the full WSJ10, NEGRA10 and CTB10 corpora. Table 1 shows that U-DOP* performs worse than UML-DOP in all cases, although the differences are small and was statistically significant only for WSJ10 using paired *t*-testing.

As explained above, the main advantage of U-DOP* over UML-DOP is that it works with a more succinct grammar extracted from the shortest derivations of HC. Table 2 shows the size of the grammar (number of rules or subtrees) of the two models for resp. Penn WSJ10, the entire Penn WSJ and the first 2 million sentences from the NANC (North American News Text) corpus which contains a total of approximately 24 million sentences from different news sources.

Model	Size of STSG for WSJ10	Size of STSG for Penn WSJ	Size of STSG for 2,000K NANC
U-DOP*	2.2×10^4	9.8×10^5	7.2×10^6
UML-DOP	1.5×10^6	8.1×10^7	5.8×10^9

Table 2. Grammar size of U-DOP* and UML-DOP for WSJ10 (7,7K sentences), WSJ (50K sentences) and the first 2,000K sentences from NANC.

Note that while U-DOP* is about 2 orders of magnitudes smaller than UML-DOP for the WSJ10, it is almost 3 orders of magnitudes smaller for the first 2 million sentences of the NANC corpus. Thus even if U-DOP* does not give the highest f-score in table 1, it is more apt to be trained on larger data sets. In fact, a well-known advantage of unsupervised methods over supervised methods is the availability of almost unlimited amounts of text. Table 2 indicates that U-DOP*'s grammar is still of manageable size even for text corpora that are (almost) two orders of magnitude larger than Penn's WSJ. The NANC corpus contains approximately 2 million WSJ sentences that do not overlap with Penn's WSJ and has been previously used by McClosky et al. (2006) in improving a supervised parser by self-training. In our experiments below we will start by mixing subsets from the NANC's WSJ data with Penn's WSJ data. Next, we will do the same with 2 million sentences from the LA Times in the NANC corpus, and finally we will mix all data together for inducing a U-DOP* model. From Penn's WSJ, we only use sections 2 to 21 for training (just as in supervised parsing) and section 23 (≤100 words) for testing, so as to compare our unsupervised results with some binarized supervised parsers.

The NANC data was first split into sentences by means of a simple discriminitive model. It was next p-o-s tagged with the the TnT tagger (Brants 2000) which was trained on the Penn Treebank such that the same tag set was used. Next, we added subsets of increasing size from the NANC p-o-s strings to the 40,000 Penn WSJ p-o-s strings. Each time the resulting corpus was split into two halfs and the shortest derivations were computed for one half by using the PCFG-reduction from the other half and vice versa. The resulting trees were used for extracting an STSG which in turn was used to parse section 23 of Penn's WSJ. Table 3 shows the results.

# sentences added	f-score by adding WSJ data	f-score by adding LA Times data
0 (baseline)	62.2	62.2
100k	64.7	63.0
250k	66.2	63.8
500k	67.9	64.1
1,000k	68.5	64.6
2,000k	69.0	64.9

Table 3. Results of U-DOP* on section 23 from Penn's WSJ by adding sentences from NANC's WSJ and NANC's LA Times

Table 3 indicates that there is a monotonous increase in f-score on the WSJ test set if NANC text is added to our training data in both cases, independent of whether the sentences come from the WSJ domain or the LA Times domain. Although the effect of adding LA Times data is weaker than adding WSJ data, it is noteworthy that the unsupervised induction of trees from the LA Times domain still improves the f-score even if the test data are from a different domain.

We also investigated the effect of adding the LA Times data to the total mix of Penn's WSJ and NANC's WSJ. Table 4 shows the results of this experiment, where the baseline of 0 sentences thus starts with the 2,040k sentences from the combined Penn-NANC WSJ data.

Sentences added from LA Times to Penn-NANC WSJ	f-score by adding LA Times data
0	69.0
100k	69.4
250k	69.9
500k	70.2
1,000k	70.4
2,000k	70.7

Table 4. Results of U-DOP* on section 23 from Penn's WSJ by mixing sentences from the combined Penn-NANC WSJ with additions from NANC's LA Times.

As seen in table 4, the f-score continues to increase even when adding LA Times data to the large combined set of Penn-NANC WSJ sentences. The highest f-score is obtained by adding 2,000k sentences, resulting in a total training set of 4,040k sentences. We believe that our result is quite promising for the future of unsupervised parsing.

In putting our best f-score in table 4 into perspective, it should be kept in mind that the gold standard trees from Penn-WSJ section 23 were binarized. It is well known that such a binarization has a negative effect on the f-score. Bod (2006) reports that an unbinarized treebank grammar achieves an average 72.3% f-score on WSJ sentences ≤ 40 words, while the binarized version achieves only 64.6% f-score. To compare U-DOP*'s results against some supervised parsers, we additionally evaluated a PCFG treebank grammar and the supervised DOP* parser using the same test set. For these supervised parsers, we employed the standard training set, i.e. Penn's WSJ sections 2-21, but only by taking the p-o-s strings as we did for our unsupervised U-DOP* model. Table 5 shows the results of this comparison.

Parser	f-score
U-DOP*	70.7
Binarized treebank PCFG	63.5
Binarized DOP*	80.3

Table 5. Comparison between the (best version of) U-DOP*, the supervised treebank PCFG and the supervised DOP* for section 23 of Penn's WSJ

As seen in table 5, U-DOP* outperforms the binarized treebank PCFG on the WSJ test set. While a similar result was obtained in Bod (2006), the absolute difference between unsupervised parsing and the treebank grammar was extremely small in Bod (2006): 1.8%, while the difference in table 5 is 7.2%, corresponding to 19.7% error reduction. Our f-score remains behind the supervised version of DOP* but the gap gets narrower as more training data is being added to U-DOP*.

5 Evaluation on unlabeled corpora in a practical application

Our experiments so far have shown that despite the addition of large amounts of unlabeled training data, U-DOP* is still outperformed by the supervised DOP* model when tested on hand-annotated corpora like the Penn Treebank. Yet it is well known that any evaluation on hand-annotated corpora unreasonably favors supervised parsers. There is thus a quest for designing an evaluation scheme that is independent of annotations. One way to go would be to compare supervised and unsupervised parsers as a syntax-based language model in a practical application such as machine translation (MT) or speech recognition.

In Bod (2007), we compared U-DOP* and DOP* in a syntax-based MT system known as Data-Oriented Translation or *DOT* (Poutsma 2000; Groves et al. 2004). The DOT model starts with a bilingual treebank where each tree pair constitutes an example translation and where translationally equivalent constituents are linked. Similar to DOP,

the DOT model uses all linked subtree pairs from the bilingual treebank to form an STSG of linked subtrees, which are used to compute the most probable translation of a target sentence given a source sentence (see Hearne and Way 2006).

What we did in Bod (2007) is to let both DOP* and U-DOP* compute the best trees directly for the *word strings* in the German-English Europarl corpus (Koehn 2005), which contains about 750,000 sentence pairs. Differently from U-DOP*, DOP* needed to be trained on annotated data, for which we used respectively the Negra and the Penn treebank. Of course, it is well-known that a supervised parser's f-score decreases if it is transferred to another domain: for example, the (non-binarized) WSJ-trained DOP model in Bod (2003) decreases from around 91% to 85.5% f-score if tested on the Brown corpus. Yet, this score is still considerably higher than the accuracy obtained by the unsupervised U-DOP model, which achieves 67.6% unlabeled f-score on Brown sentences. Our main question of interest is in how far this difference in accuracy on hand-annotated corpora carries over when tested in the context of a concrete application like MT. This is not a trivial question, since U-DOP* learns 'constituents' for word sequences such as *Ich möchte* ("I would like to") and *There are* (Bod 2007), which are usually hand-annotated as non-constituents. While U-DOP* is punished for this 'incorrect' prediction if evaluated on the Penn Treebank, it may be rewarded for this prediction if evaluated in the context of machine translation using the Bleu score (Papineni et al. 2002). Thus similar to Chiang (2005), U-DOP can discover non-syntactic phrases, or simply "phrases", which are typically neglected by linguistically syntax-based MT systems. At the same time, U-DOP* can also learn discontiguous constituents that are neglected by phrase-based MT systems (Koehn et al. 2003).

In our experiments, we used both U-DOP* and DOP* to predict the best trees for the German-English Europarl corpus. Next, we assigned links between each two nodes in the respective trees for each sentence pair. For a 2,000 sentence test set from a different part of the Europarl corpus we computed the most probable target sentence (using Viterbi *n* best). The Bleu score was used to measure translation accuracy, calculated by the NIST script with its default settings. As a baseline we compared our results with the publicly available phrase-based system Pharaoh (Koehn et al. 2003), using the default feature set. Table 6 shows for each system the Bleu score together with a description of the productive units. 'U-DOT' refers to 'Unsupervised DOT' based on U-DOP*, while DOT is based on DOP*.

System	Productive Units	Bleu-score
U-DOT / U-DOP*	Constituents and Phrases	0.280
DOT / DOP*	Constituents only	0.221
Pharaoh	Phrases only	0.251

Table 6. Comparing U-DOP* and DOP* in syntax-based MT on the German-English Europarl corpus against the Pharaoh system.

The table shows that the unsupervised U-DOT model outperforms the supervised DOT model with 0.059. Using Zhang's significance tester (Zhang et al. 2004), it turns out that this difference is statistically significant ($p < 0.001$). Also the difference between U-DOT and the baseline Pharaoh is statistically significant ($p < 0.008$). Thus even if supervised parsers like DOP* outperform unsupervised parsers like U-DOP* on hand-parsed data with >10%, the same supervised parser is outperformed by the unsupervised parser if tested in an MT application. Evidently, U-DOP's capacity to capture both constituents and phrases pays off in a concrete application and shows the shortcomings of models that only allow for either constituents (such as linguistically syntax-based MT) or phrases (such as phrase-based MT). In Bod (2007) we also show that U-DOT obtains virtually the same Bleu score as Pharaoh after eliminating subtrees with discontiguous yields.

6 Conclusion: future of supervised parsing

In this paper we have shown that the accuracy of unsupervised parsing under U-DOP* continues to grow when enlarging the training set with additional data. However, except for the simple treebank PCFG, U-DOP* scores worse than supervised parsers if evaluated on hand-annotated data. At the same time U-DOP* significantly outperforms the supervised DOP* if evaluated in a practical application like MT. We argued that this can be explained by the fact that U-DOP learns

both constituents and (non-syntactic) phrases while supervised parsers learn constituents only.

What should we learn from these results? We believe that parsing, when separated from a task-based application, is mainly an academic exercise. If we only want to mimick a treebank or implement a linguistically motivated grammar, then supervised, grammar-based parsers are preferred to unsupervised parsers. But if we want to improve a practical application with a syntax-based language model, then an unsupervised parser like U-DOP* might be superior.

The problem with most supervised (and semi-supervised) parsers is their rigid notion of constituent which excludes 'constituents' like the German *Ich möchte* or the French *Il y a*. Instead, it has become increasingly clear that the notion of constituent is a fluid which may sometimes be in agreement with traditional syntax, but which may just as well be in opposition to it. Any sequence of words can be a unit of combination, including non-contiguous word sequences like *closest* X *to* Y. A parser which does not allow for this fluidity may be of limited use as a language model. Since supervised parsers seem to stick to categorical notions of constituent, we believe that in the field of syntax-based language models the end of supervised parsing has come in sight.

Acknowledgements

Thanks to Willem Zuidema and three anonymous reviewers for useful comments and suggestions on the future of supervised parsing.

References

Billot, S. and B. Lang, 1989. The Structure of Shared Forests in Ambiguous Parsing. In *ACL 1989*.

Bod, R. 1998. *Beyond Grammar: An Experience-Based Theory of Language*, CSLI Publications.

Bod, R. Parsing with the Shortest Derivation. In *COLING 2000*, Saarbruecken.

Bod, R. 2003. An efficient implementation of a new DOP model. In *EACL 2003*, Budapest.

Bod, R. 2006. An All-Subtrees Approach to Unsupervised Parsing. In *ACL-COLING 2006*, Sydney.

Bod, R. 2007. Unsupervised Syntax-Based Machine Translation. Submitted for publication.

Brants, T. 2000. TnT - A Statistical Part-of-Speech Tagger. In *ANLP 2000*.

Chiang, D. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation. In *ACL 2005*, Ann Arbor.

Clark, A. 2001. Unsupervised induction of stochastic context-free grammars using distributional clustering. In *CONLL 2001*.

Goodman, J. 2003. Efficient algorithms for the DOP model. In R. Bod, R. Scha and K. Sima'an (eds.). *Data-Oriented Parsing*, CSLI Publications.

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

Groves, D., M. Hearne and A. Way, 2004. Robust Sub-Sentential Alignment of Phrase-Structure Trees. In *COLING 2004*, Geneva.

Hearne, M and A. Way, 2006. Disambiguation Strategies for Data-Oriented Translation. *Proceedings of the 11th Conference of the European Association for Machine Translation*, Oslo.

Henderson, J. 2004. Discriminative training of a neural network statistical parser. In *ACL 2004,* Barcelona.

Huang, L. and D. Chiang 2005. Better *k*-best parsing. In *IWPT 2005*, Vancouver.

Johnson, M. 2002. The DOP estimation method is biased and inconsistent. *Computational Linguistics* 28, 71-76.

Klein, D. and C. Manning 2002. A general constituent-context model for improved grammar induction. In *ACL 2002*, Philadelphia.

Klein, D. and C. Manning 2004. Corpus-based induction of syntactic structure: models of dependency and constituency. *ACL 2004*, Barcelona.

Koehn, P., Och, F. J., and Marcu, D. 2003. Statistical phrase based translation. In *HLT-NAACL 2003*.

Koehn, P. 2005. Europarl: a parallel corpus for statistical machine translation. In *MT Summit 2005*.

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

Poutsma, A. 2000. Data-Oriented Translation. In *COLING 2000*, Saarbruecken.

Schütze, H. 1995. Distributional part-of-speech tagging. In *ACL 1995*, Dublin.

Sima'an, K. 1996. Computational complexity of probabilistic disambiguation by means of tree grammars. In *COLING 1996*, Copenhagen.

Steedman, M. M. Osborne, A. Sarkar, S. Clark, R. Hwa, J. Hockenmaier, P. Ruhlen, S. Baker, and J. Crim. 2003. Bootstrapping statistical parsers from small datasets. In *EACL 2003*, Budapest.

van Zaanen, M. 2000. ABL: Alignment-Based Learning. In *COLING 2000*, Saarbrücken.

Zhang, Y., S. Vogel and A. Waibel, 2004. Interpreting BLEU/NIST scores: How much improvement do we need to have a better system? *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC).*

Zollmann, A. and K. Sima'an 2005. A consistent and efficient estimator for data-oriented parsing. *Journal of Automata, Languages and Combinatorics*, Vol. 10 (2005) Number 2/3, 367-388.

An Ensemble Method for Selection of High Quality Parses

Roi Reichart
ICNC
Hebrew University of Jerusalem
roiri@cs.huji.ac.il

Ari Rappoport
Institute of Computer Science
Hebrew University of Jerusalem
arir@cs.huji.ac.il

Abstract

While the average performance of statistical parsers gradually improves, they still attach to many sentences annotations of rather low quality. The number of such sentences grows when the training and test data are taken from different domains, which is the case for major web applications such as information retrieval and question answering.

In this paper we present a *Sample Ensemble Parse Assessment (SEPA)* algorithm for detecting parse quality. We use a function of the agreement among several copies of a parser, each of which trained on a different sample from the training data, to assess parse quality. We experimented with both generative and reranking parsers (Collins, Charniak and Johnson respectively). We show superior results over several baselines, both when the training and test data are from the same domain and when they are from different domains. For a test setting used by previous work, we show an error reduction of 31% as opposed to their 20%.

1 Introduction

Many algorithms for major NLP applications such as information extraction (IE) and question answering (QA) utilize the output of statistical parsers (see (Yates et al., 2006)). While the average performance of statistical parsers gradually improves, the quality of many of the parses they produce is too low for applications. When the training and test data are taken from different domains (the *parser adaptation* scenario) the ratio of such low quality parses becomes even higher. Figure 1 demonstrates these phenomena for two leading models, Collins (1999) model 2, a generative model, and Charniak and Johnson (2005), a reranking model. The parser adaptation scenario is the rule rather than the exception for QA and IE systems, because these usually operate over the highly variable Web, making it very difficult to create a representative corpus for manual annotation. Medium quality parses may seriously harm the performance of such systems.

In this paper we address the problem of assessing parse quality, using a *Sample Ensemble Parse Assessment (SEPA)* algorithm. We use the level of agreement among several copies of a parser, each of which trained on a different sample from the training data, to predict the quality of a parse. The algorithm does not assume uniformity of training and test data, and is thus suitable to web-based applications such as QA and IE.

Generative statistical parsers compute a probability $p(a, s)$ for each sentence annotation, so the immediate technique that comes to mind for assessing parse quality is to simply use $p(a, s)$. Another seemingly trivial method is to assume that shorter sentences would be parsed better than longer ones. However, these techniques produce results that are far from optimal. In Section 5 we show the superiority of our method over these and other baselines.

Surprisingly, as far as we know there is only one previous work explicitly addressing this problem (Yates et al., 2006). Their WOODWARD algorithm filters out high quality parses by performing seman-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 408–415,
Prague, Czech Republic, June 2007.

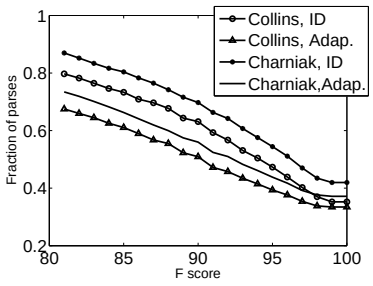


Figure 1: F-score vs. the fraction of parses whose f-score is at least that f-score. For the in-domain scenario, the parsers are tested on sec 23 of the WSJ Penn Treebank. For the parser adaptation scenario, they are tested on the Brown test section. In both cases they are trained on sections 2-21 of WSJ.

tic analysis. The present paper provides a detailed comparison between the two algorithms, showing both that SEPA produces superior results and that it operates under less restrictive conditions.

We experiment with both the generative parsing model number 2 of Collins (1999) and the reranking parser of Charniak and Johnson (2005), both when the training and test data belong to the same domain (the *in-domain* scenario) and in the parser adaptation scenario. In all four cases, we show substantial improvement over the baselines. The present paper is the first to use a reranking parser and the first to address the adaptation scenario for this problem.

Section 2 discusses relevant previous work, Section 3 describes the SEPA algorithm, Sections 4 and 5 present the experimental setup and results, and Section 6 discusses certain aspects of these results and compares SEPA to WOODWARD.

2 Related Work

The only previous work we are aware of that explicitly addressed the problem of detecting high quality parses in the output of statistical parsers is (Yates et al., 2006). Based on the observation that incorrect parses often result in implausible semantic interpretations of sentences, they designed the WOODWARD filtering system. It first maps the parse produced by the parser to a logic-based representation (relational conjunction (RC)) and then employs four methods for semantically analyzing whether a conjunct in the RC is likely to be reasonable. The filters use semantic information obtained from the Web. Measuring errors using filter f-score (see Section 3) and using the Collins generative model, WOODWARD reduces errors by 67% on a set of TREC questions and by 20% on a set of a 100 WSJ sentences. Section 5 provides a detailed comparison with our algorithm.

Reranking algorithms (Koo and Collins, 2005; Charniak and Johnson, 2005) search the list of best parses output by a generative parser to find a parse of higher quality than the parse selected by the generative parser. Thus, these algorithms in effect assess parse quality using syntactic and lexical features. The SEPA algorithm does not use such features, and is successful in detecting high quality parses even when working on the output of a reranker. Reranking and SEPA are thus relatively independent.

Bagging (Breiman, 1996) uses an ensemble of instances of a model, each trained on a sample of the training data[1]. Bagging was suggested in order to enhance classifiers; the classification outcome was determined using a majority vote among the models. In NLP, bagging was used for active learning for text classification (Argamon-Engelson and Dagan, 1999; McCallum and Nigam, 1998). Specifically in parsing, (Henderson and Brill, 2000) applied a constituent level voting scheme to an ensemble of bagged models to increase parser performance, and (Becker and Osborne, 2005) suggested an active learning technique in which the agreement among an ensemble of bagged parsers is used to predict examples valuable for human annotation. They reported experiments with small training sets only (up to 5,000 sentences), and their agreement function is very different from ours. Both works experimented with generative parsing models only.

Ngai and Yarowsky (2000) used an ensemble based on bagging and partitioning for active learning for base NP chunking. They select top items without any graded assessment, and their f-complement function, which slightly resembles our MF (see the next section), is applied to the output of a classifier, while our function is applied to structured output. A survey of several papers dealing with mapping

[1]Each sample is created by sampling, with replacement, L examples from the training pool, where L is the size of the training pool. Conversely, each of our samples is smaller than the training set, and is created by sampling without replacement. See Section 3 ('regarding S') for a discussion of this issue.

predictors in classifiers' output to posterior probabilities is given in (Caruana and Niculescu-Mizil, 2006). As far as we know, the application of a sample based parser ensemble for assessing parse quality is novel.

Many IE and QA systems rely on the output of parsers (Kwok et al., 2001; Attardi et al., 2001; Moldovan et al., 2003). The latter tries to address incorrect parses using complex relaxation methods. Knowing the quality of a parse could greatly improve the performance of such systems.

3 The Sample Ensemble Parse Assessment (SEPA) Algorithm

In this section we detail our parse assessment algorithm. Its input consists of a parsing algorithm A, an annotated training set TR, and an unannotated test set TE. The output provides, for each test sentence, the parse generated for it by A when trained on the full training set, and a grade assessing the parse's quality, on a continuous scale between 0 to 100. Applications are then free to select a sentence subset that suits their needs using our grades, e.g. by keeping only high-quality parses, or by removing low-quality parses and keeping the rest. The algorithm has the following stages:

1. Choose N random samples of size S from the training set TR. Each sample is selected without replacement.
2. Train N copies of the parsing algorithm A, each with one of the samples.
3. Parse the test set with each of the N models.
4. For each test sentence, compute the value of an agreement function F between the models.
5. Sort the test set according to F's value.

The algorithm uses the level of agreement among several copies of a parser, each trained on a different sample from the training data, to predict the quality of a parse. The higher the agreement, the higher the quality of the parse. Our approach assumes that if the parameters of the model are well designed to annotate a sentence with a high quality parse, then it is likely that the model will output the same (or a highly similar) parse even if the training data is somewhat changed. In other words, we rely on the stability of the parameters of statistical parsers. Although this is not always the case, our results confirm that strong correlation between agreement and parse quality does exist.

We explored several agreement functions. The one that showed the best results is *Mean F-score (MF)*[2], defined as follows. Denote the models by $m_1 \dots m_N$, and the parse provided by m_i for sentence s as $m_i(s)$. We randomly choose a model m_l, and compute

$$MF(s) = \frac{1}{N-1} \sum_{i \in [1 \dots N], i \neq l} fscore(m_i, m_l) \quad (1)$$

We use two measures to evaluate the quality of SEPA grades. Both measures are defined using a threshold parameter T, addressing only sentences whose SEPA grades are not smaller than T. We refer to these sentences as *T-sentences*.

The first measure is the average f-score of the parses of T-sentences. Note that we compute the f-score of each of the selected sentences and then average the results. This stands in contrast to the way f-score is ordinarily calculated, by computing the labeled precision and recall of the constituents in the whole set and using these as the arguments of the f-score equation. The ordinary f-score is computed that way mostly in order to overcome the fact that sentences differ in length. However, for applications such as IE and QA, which work at the single sentence level and which might reach erroneous decision due to an inaccurate parse, normalizing over sentence lengths is less of a factor. For this reason, in this paper we present detailed graphs for the average f-score. For completeness, Table 4 also provides some of the results using the ordinary f-score.

The second measure is a generalization of the filter f-score measure suggested by Yates et al. (2006). They define *filter precision* as the ratio of correctly parsed sentences in the *filtered set* (the set the algorithm choose) to total sentences in the filtered set and *filter recall* as the ratio of correctly parsed sentences in the filtered set to correctly parsed sentences in the

[2]Recall that sentence f-score is defined as: $f = \frac{2 \times P \times R}{P+R}$, where P and R are the labeled precision and recall of the constituents in the sentence relative to another parse.

whole set of sentences parsed by the parser (*unfiltered set* or *test set*). Correctly parsed sentences are sentences whose parse got f-score of 100%.

Since requiring a 100% may be too restrictive, we generalize this measure to *filter f-score with parameter* k. In our measure, the filter recall and precision are calculated with regard to sentences that get an f-score of k or more, rather than to correctly parsed sentences. Filtered f-score is thus a special case of our filtered f-score, with parameter 100.

We now discuss the effect of the number of models N and the sample size S. The discussion is based on experiments (using development data, see Section 4) in which all the parameters are fixed except for the parameter in question, using our development sections.

Regarding N (see Figure 2): As the number of models increases, the number of T-sentences selected by SEPA decreases and their quality improves, in terms of both average f-score and filter f-score (with $k = 100$). The fact that more models trained on different samples of the training data agree on the syntactic annotation of a sentence implies that this syntactic pattern is less sensitive to perturbations in the training data. The number of such sentences is small and it is likely the parser will correctly annotate them. The smaller T-set size leads to a decrease in filter recall, while the better quality leads to an increase in filter precision. Since the increase in filter precision is sharper than the decrease in filter recall, filter f-score increases with the number of models N.

Regarding S[3]: As the sample size increases, the number of T-sentences increases, and their quality degrades in terms of average f-score but improves in terms of filter f-score (again, with parameter $k = 100$). The overlap among smaller samples is small and the data they supply is sparse. If several models trained on such samples attach to a sentence the same parse, this syntactic pattern must be very prominent in the training data. The number of such sentences is small and it is likely that the parser will correctly annotate them. Therefore smaller sample size leads to smaller T-sets with high average f-score. As the sample size increases, the T-set becomes larger but the average f-score of a parse

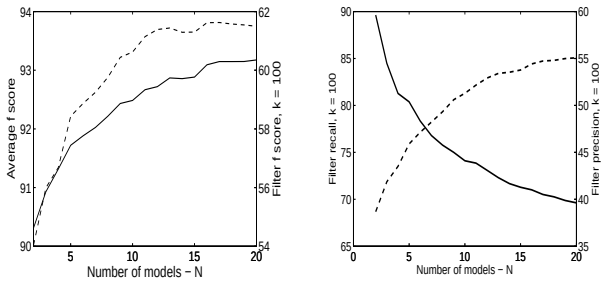


Figure 2: The effect of the number of models N on SEPA (Collins' model). The scenario is in-domain, sample size $S = 33,000$ and $T = 100$. We see: average f-score of T-sentences (left, solid curve and left y-axis), filter f-score with $k = 100$ (left, dashed curve and right y-axis), filter recall with $k = 100$ (right, solid curve and left y-axis), and filter precision with $k = 100$ (right, dashed curve and right y-axis).

decreases. The larger T-set size leads to increase in filter recall, while the lower average quality leads to decrease in filter precision. Since the increase in filter recall is sharper than the decrease in filter precision, the result is that filter f-score increases with the sample size S.

This discussion demonstrates the importance of using both average f-score and filter f-score, since the two measures reflect characteristics of the selected sample that are not necessarily highly (or positively) correlated.

4 Experimental Setup

We performed experiments with two parsing models, the Collins (1999) generative model number 2 and the Charniak and Johnson (2005) reranking model. For the first we used a reimplementation (**?**). We performed experiments with each model in two scenarios, in-domain and parser adaptation. In both experiments the training data are sections 02-21 of the WSJ PennTreebank (about 40K sentences). In the in-domain experiment the test data is section 23 (2416 sentences) of WSJ and in the parser adaptation scenario the test data is Brown test section (2424 sentences). Development sections are WSJ section 00 for the in-domain scenario (1981 sentences) and Brown development section for the adaptation scenario (2424 sentences). Following

[3]Graphs are not shown due to lack of space.

(Gildea, 2001), the Brown test and development sections consist of 10% of Brown sentences (the 9th and 10th of each 10 consecutive sentences in the development and test sections respectively).

We performed experiments with many configurations of the parameters *N* (number of models), *S* (sample size) and *F* (agreement function). Due to space limitations we describe only experiments where the values of the parameters *N*, *S* and *F* are fixed (*F* is MF, N and S are given in Section 5) and the threshold parameter T is changed.

5 Results

We first explore the quality of the selected set in terms of average f-score. In Section 3 we reported that the quality of a selected T-set of parses increases as the number of models N increases and sample size S decreases. We therefore show the results for relatively high N (20) and relatively low S (13,000, which is about a third of the training set). Denote the cardinality of the set selected by SEPA by n (it is actually a function of T but we omit the T in order to simplify notations).

We use several baseline models. The first, *confidence baseline (CB)*, contains the n sentences having the highest parser assigned probability (when trained on the whole training set). The second, *minimum length (ML)*, contains the n shortest sentences in the test set. Since many times it is easier to parse short sentences, a trivial way to increase the average f-score measure of a set is simply to select short sentences. The third, following (Yates et al., 2006), is *maximum recall (MR)*. MR simply predicts that all test set sentences should be contained in the selected T-set. The output set of this model gets filter recall of 1 for any k value, but its precision is lower. The MR baseline is not relevant to the average f-score measure, because it selects all of the sentences in a set, which leads to the same average as a random selection (see below). In order to minimize visual clutter, for the filter f-score measure we use the maximum recall (MR) baseline rather than the minimum length (ML) baseline, since the former outperforms the latter. Thus, ML is only shown for the average f-score measure. We have also experimented with a random baseline model (containing n randomly selected test sentences), whose results are the worst and which is shown for reference.

Readers of this section may get confused between the agreement threshold parameter T and the parameter k of the filter f-score measure. Please note: as to T, SEPA sorts the test set by the values of the agreement function. One can then select only sentences whose agreement score is at least T. T's values are on a continuous scale from 0 to 100. As to k, the filter f-score measure gives a grade. This grade combines three values: (1) the number of sentences in the set (selected by an algorithm) whose f-score relative to the gold standard parse is at least k, (2) the size of the selected set, and (3) the total number of sentences with such a parse in the whole test set. We did not introduce separate notations for these values.

Figure 3 (top) shows average f-score results where SEPA is applied to Collins' generative model in the in-domain (left) and adaptation (middle) scenarios. SEPA outperforms the baselines for all values of the agreement threshold parameter T. Furthermore, as T increases, not only does the SEPA set quality increase, but the quality differences between this set and the baseline sets increases as well. The graphs on the right show the number of sentences in the sets selected by SEPA for each T value. As expected, this number decreases as T increases.

Figure 3 (bottom) shows the same pattern of results for the Charniak reranking parser in the in-domain (left) and adaptation (middle) scenarios. We see that the effects of the reranker and SEPA are relatively independent. Even after some of the errors of the generative model were corrected by the reranker by selecting parses of higher quality among the 50-best, SEPA can detect parses of high quality from the set of parsed sentences.

To explore the quality of the selected set in terms of filter f-score, we recall that the quality of a selected set of parses increases as both the number of models N and the sample size S increase, and with T. Therefore, for $k = 85 \ldots 100$ we show the value of filter f-score with parameter k when the parameters configuration is a relatively high N (20), relatively high S (33,000, which are about 80% of the training set), and the highest T (100).

Figure 4 (top) shows filter f-score results for Collins' generative model in the in-domain (left) and adaptation (middle) scenarios. As these graphs show, SEPA outperforms CB and random for all val-

ues of the filter f-score parameter k, and outperforms the MR baseline where the value of k is 95 or more. Although for small k values MR gets a higher f-score than SEPA, the filter precision of SEPA is much higher (right, shown for adaptation. The in-domain pattern is similar and not shown). This stems from the definition of the MR baseline, which simply predicts any sentence to be in the selected set. Furthermore, since the selected set is meant to be the input for systems that require high quality parses, what matters most is that SEPA outperforms the MR baseline at the high k ranges.

Figure 4 (bottom) shows the same pattern of results for the Charniak reranking parser in the in-domain (left) and adaptation (middle) scenarios. As for the average f-score measure, it demonstrates that the effects of the reranker and SEPA algorithm are relatively independent.

Tables 1 and 2 show the error reduction achieved by SEPA for the filter f-score measure with parameters $k = 95, 97, 100$ (Table 1) and for the average f-score measure with several SEPA agreement threshold (T) values (Table 2) . The error reductions achieved by SEPA for both measures are substantial.

Table 3 compares SEPA and WOODWARD on the exact same test set used by (Yates et al., 2006) (taken from WSJ sec 23). SEPA achieves error reduction of 31% over the MR baseline on this set, compared to only 20% achieved by WOODWARD. Not shown in the table, in terms of ordinary f-score WOODWARD achieves error reduction of 37% while SEPA achieves 43%. These numbers were the only ones reported in (Yates et al., 2006).

For completeness of reference, Table 4 shows the superiority of SEPA over CB in terms of the usual f-score measure used by the parsing community (numbers are counted for constituents first). Results for other baselines are even more impressive. The configuration is similar to that of Figure 3.

	Filter f-score					
	In-domain			Adaptation		
k value	95	97	100	95	97	100
Coll. MR	3.5	20.1	29.2	22.8	29.8	33.6
Coll. CB	11.6	11.7	3.4	14.2	9.9	7.4
Char. MR	1.35	13.6	23.44	21.9	30	32.5
Char. CB	21.9	16.8	11.9	25	20.2	16.2

Table 1: Error reduction in the filter f-score measure obtained by SEPA with Collins' (top two lines) and Charniak's (bottom two lines) model, in the two scenarios (in-domain and adaptation), vs. the maximum recall (MR lines 1 and 3) and confidence (CB, lines 2 and 4) baselines, using $N = 20, T = 100$ and $S = 33,000$. Shown are parameter values $k = 95, 97, 100$. Error reduction numbers were computed by $100 \times (fscoreSEPA - fscorebaseline)/(1 - fscorebaseline)$.

	Average f-score					
	In-domain			Adaptation		
T	95	97	100	95	97	100
Coll. ML	32.6	37.2	60.8	46.8	52.7	70.7
Coll. CB	26.5	31.4	53.9	46.9	53.6	70
Char. ML	25.1	33.2	58.5	46.9	58.4	77.1
Char. CB	20.4	30	52	44.4	55.5	73.5

Table 2: Error reduction in the average f-score measure obtained by SEPA with Collins (top two lines) and Charniak (bottom two lines) model, in the two scenarios (in-domain and adaptation), vs. the minimum length (ML lines 1 and 3) and confidence (CB, lines 2 and 4) baselines, using $N = 20$ and $S = 13,000$. Shown are agreement threhsold parameter values $T = 95, 97, 100$. Error reduction numbers were computed by $100 \times (fscoreSEPA - fscorebaseline)/(1 - fscorebaseline)$.

	SEPA	WOODWARD	CB
ER	**31%**	20%	-31%

Table 3: Error reduction compared to the MR baseline, measured by filter f-score with parameter 100. The data is the WSJ sec 23 test set usd by (Yates et al., 2006). All three methods use Collins' model. SEPA uses $N = 20, S = 33,000, T = 100$.

6 Discussion

In this paper we introduced SEPA, a novel algorithm for assessing parse quality in the output of a statistical parser. SEPA is the first algorithm shown to be successful when a reranking parser is considered, even though such models use a reranker to detect and fix some of the errors made by the base generative model. WOODWARD, the only previously suggested algorithm for this problem, was tested with Collins' generative model only. Furthermore, this is the first time that an algorithm for this problem succeeds in a domain adaptation scenario, regardless of

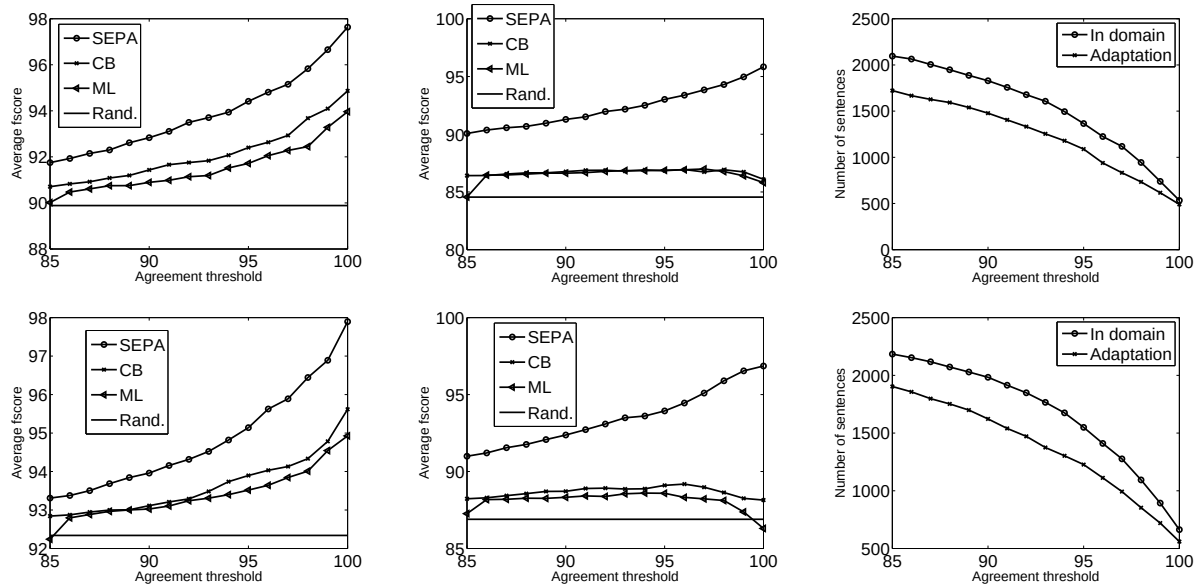


Figure 3: Agreement threshold T vs. average f-score (left and middle) and number of sentences in the selected set (right), for SEPA with Collins' generative model (top) and the Charniak reranking model (bottom). SEPA parameters are $S = 13,000, N = 20$. In both rows, SEPA results for the in-domain (left) and adaptation (middle) scenarios are compared to the confidence (CB) and minimum length (ML) baselines. The graphs on the right show the number of sentences in the selected set for both scenarios.

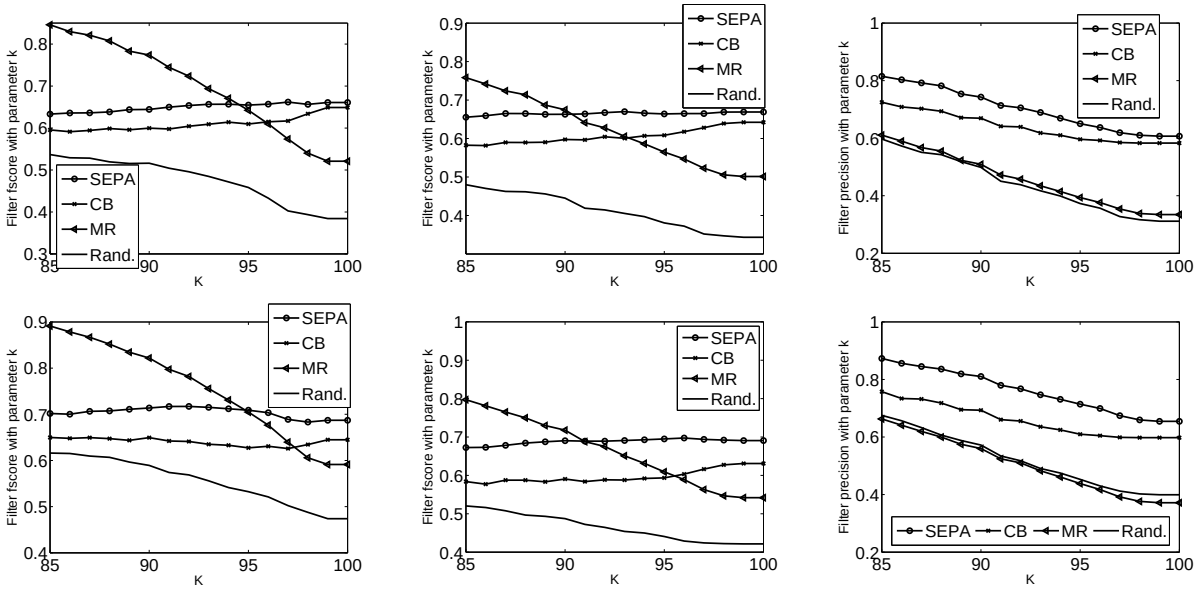


Figure 4: Parameter k vs. filter f-score (left and middle) and filter precision (right) with that parameter, for SEPA with Collins' generative model (top) and the Charniak reranking model (bottom). SEPA parameters are $S = 33,000, N = 20, T = 100$. In both rows, results for the in-domain (left) and adaptation (middle) scenarios. In two leftmost graphs, the performance of the algorithm is compared to the confidence baseline (CB) and maximum recall (MR). The graphs on the right compare the filter precision of SEPA with that of the MR and CB baselines.

the parsing model. In the Web environment this is the common situation.

The WSJ and Brown experiments performed with SEPA are much broader than those performed with WOODWARD, considering all sentences of WSJ sec 23 and Brown test section rather than a subset of carefully selected sentences from WSJ sec 23. However, we did not perform a TREC experiment, as (Yates et al., 2006) did. Our WSJ and Brown results outperformed several baselines. Moreover, WSJ (or Brown) sentences that contain conjunctions were avoided in the experiments of (Yates et al., 2006). We have verified that our algorithm shows substantial error reduction over the baselines for this type of sentences (in the ranges $13 - 46\%$ for the filter f-score with $k = 100$, and $30 - 60\%$ for the average f-score).

As Table 3 shows, on a WSJ sec 23 test set similar to that used by (Yates et al., 2006), SEPA achieves 31% error reduction compared to 20% of WOODWARD.

WOODWARD works under several assumptions. Specifically, it requires a corpus whose content overlaps at least in part with the content of the parsed sentences. This corpus is used to extract semantically related statistics for its filters. Furthermore, the filters of this algorithm (except of the QA filter) are focused on verb and preposition relations. Thus, it is more natural for it to deal with mistakes contained in such relations. This is reflected in the WSJ based test set on which it is tested. SEPA does not make any of these assumptions. It does not use any external information source and is shown to select high quality parses from diverse sets.

	In-domain		Adaptation	
	F	ER	F	ER
SEPA Collins	97.09	44.36%	95.38	66.38%
CB Collins	94.77	–	86.3	–
SEPA Charniak	97.21	35.69%	96.3	54.66%
CB Charniak	95.6	–	91.84	–

Table 4: SEPA error reduction vs. the CB baseline in the in-domain and adaptation scenarios, using the traditional f-score of the parsing literature. $N = 20, S = 13,000, T = 100$.

For future work, integrating SEPA into the reranking process seems a promising direction for enhancing overall parser performance.

Acknowledgement. We would like to thank Dan Roth for his constructive comments on this paper.

References

Shlomo Argamon-Engelson and Ido Dagan, 1996. committee-based sample selection for probabilistic classifiers. *Journal of Artificial Intelligence Research*, 11:335–360.

Giuseppe Attardi, Antonio Cisternino, Francesco Formica, Maria Simi and Alessandro Tommasi, 2001. PiQASso: Pisa question answering system. *TREC '01*.

Markus Becker and Miles Osborne, 2005. A two-stage method for active learning of statistical grammars. *IJCAI '05*.

Daniel Bikel, 2004. *Code developed at University of Pennsylvania*. http://www.cis.upenn.edu.bikel.

Leo Breiman, 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.

Rich Caruana and Alexandru Niculescu-Mizil, 2006. An empirical comparison of supervised learning algorithms. *ICML '06*.

Eugene Charniak and Mark Johnson, 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. *ACL '05*.

Michael Collins, 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania.

Daniel Gildea, 2001. Corpus variation and parser performance. *EMNLP '01*.

John C. Henderson and Eric Brill, 2000. Bagging and boosting a treebank parser. *NAACL '00*.

Terry Koo and Michael Collins, 2005. Hidden-variable models for discriminative reranking. *EMNLP '05*.

Cody Kwok, Oren Etzioni and Daniel S. Weld, 2001. Scaling question answering to the web. *WWW '01*.

Andrew McCallum and Kamal Nigam, 1998. Employing EM and pool-based active learning for text classification. *ICML '98*.

Dan Moldovan, Christine Clark, Sanda Harabagiu and Steve Maiorano, 2003. Cogex: A logic prover for question answering. *HLT-NAACL '03*.

Grace Ngai and David Yarowsky, 2000. Rule writing or annotation: cost-efficient resource usage for base noun phrase chunking. *ACL '00*.

Alexander Yates, Stefan Schoenmackers and Oren Etzioni, 2006. Detecting parser errors using web-based semantic filters. *EMNLP '06*.

Opinion Mining Using Econometrics: A Case Study on Reputation Systems

Anindya Ghose **Panagiotis G. Ipeirotis** **Arun Sundararajan**

Department of Information, Operations, and Management Sciences
Leonard N. Stern School of Business, New York University
{aghose,panos,arun}@stern.nyu.edu

Abstract

Deriving the polarity and strength of opinions is an important research topic, attracting significant attention over the last few years. In this work, to measure the strength and polarity of an opinion, we consider the *economic context* in which the opinion is evaluated, instead of using human annotators or linguistic resources. We rely on the fact that text in on-line systems influences the behavior of humans and this effect can be observed using some easy-to-measure economic variables, such as revenues or product prices. By reversing the logic, we infer the semantic orientation and strength of an opinion by tracing the changes in the associated economic variable. In effect, we use econometrics to identify the "economic value of text" and assign a "dollar value" to each opinion phrase, measuring sentiment effectively and without the need for manual labeling. We argue that by interpreting opinions using econometrics, we have the first *objective*, *quantifiable*, and *context-sensitive* evaluation of opinions. We make the discussion concrete by presenting results on the reputation system of Amazon.com. We show that user feedback affects the pricing power of merchants and by measuring their pricing power we can infer the polarity and strength of the underlying feedback postings.

1 Introduction

A significant number of websites today allow users to post articles where they express opinions about products, firms, people, and so on. For example, users on Amazom.com post reviews about products they bought and users on eBay.com post feedback describing their experiences with sellers. The goal of opinion mining systems is to identify such pieces of the text that express opinions (Breck et al., 2007; König and Brill, 2006) and then measure the polarity and strength of the expressed opinions. While intuitively the task seems straightforward, there are multiple challenges involved.

- What makes an opinion positive or negative? Is there an *objective* measure for this task?
- How can we rank opinions according to their strength? Can we define an *objective* measure for ranking opinions?
- How does the *context* change the polarity and strength of an opinion and how can we take the context into consideration?

To evaluate the polarity and strength of opinions, most of the existing approaches rely either on training from human-annotated data (Hatzivassiloglou and McKeown, 1997), or use linguistic resources (Hu and Liu, 2004; Kim and Hovy, 2004) like WordNet, or rely on co-occurrence statistics (Turney, 2002) between words that are unambiguously positive (e.g., "excellent") and unambiguously negative (e.g., "horrible"). Finally, other approaches rely on reviews with numeric ratings from websites (Pang and Lee, 2002; Dave et al., 2003; Pang and Lee, 2004; Cui et al., 2006) and train (semi-)supervised learning algorithms to classify reviews as positive or negative, or in more fine-grained scales (Pang and Lee, 2005; Wilson et al., 2006). Implicitly, the supervised learning techniques assume that numeric ratings fully encapsulate the sentiment of the review.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 416–423,
Prague, Czech Republic, June 2007.

In this paper, we take a different approach and instead consider the *economic context* in which an opinion is evaluated. We observe that the text in on-line systems influence the behavior of the readers. This effect can be measured by observing some easy-to-measure *economic variable*, such as product prices. For instance, online merchants on eBay with "positive" feedback can sell products for higher prices than competitors with "negative" evaluations. Therefore, each of these (positive or negative) evaluations has a (positive or negative) effect on the prices that the merchant can charge. For example, everything else being equal, a seller with "speedy" delivery may be able to charge $10 more than a seller with "slow" delivery. Using this information, we can conclude that "speedy" is better than "slow" when applied to "delivery" and their difference is $10. Thus, we can infer the semantic orientation and the strength of an evaluation from the changes in the observed economic variable. Following this idea, we use techniques from econometrics to identify the "*economic value of text*" and assign a "dollar value" to each text snippet, measuring sentiment strength and polarity effectively and without the need for labeling or any other resource.

We argue that by interpreting opinions within an econometric framework, we have the first *objective* and *context-sensitive* evaluation of opinions. For example, consider the comment "good packaging," posted by a buyer to evaluate a merchant. This comment would have been considered unambiguously positive by the existing opinion mining systems. We observed, though, that within electronic markets, such as eBay, a posting that contains the words "good packaging" has actually *negative* effect on the power of a merchant to charge higher prices. This surprising effect reflects the nature of the comments in online marketplaces: buyers tend to use superlatives and highly enthusiastic language to praise a good merchant, and a lukewarm "good packaging" is interpreted as negative. By introducing the econometric interpretation of opinions we can effortlessly capture such challenging scenarios, something that is impossible to achieve with the existing approaches.

We focus our paper on reputation systems in electronic markets and we examine the effect of opinions on the pricing power of merchants in the marketplace of Amazon.com. (We discuss more applications in Section 7.) We demonstrate the value of our technique using a dataset with 9,500 transactions that took place over 180 days. We show that textual feedback affects the power of merchants to charge higher prices than the competition, for the same product, and still make a sale. We then reverse the logic and determine the contribution of each comment in the pricing power of a merchant. Thus, we discover the polarity and strength of each evaluation without the need for human annotation or any other form of linguistic resource.

The structure of the rest of the paper is as follows. Section 2 gives the basic background on reputation systems. Section 3 describes our methodology for constructing the data set that we use in our experiments. Section 4 shows how we combine established techniques from econometrics with text mining techniques to identify the strength and polarity of the posted feedback evaluations. Section 5 presents the experimental evaluations of our techniques. Finally, Section 6 discusses related work and Section 7 discusses further applications and concludes the paper.

2 Reputation Systems and Price Premiums

When buyers purchase products in an electronic market, they assess and pay not only for the product they wish to purchase but for a set of fulfillment characteristics as well, e.g., packaging, delivery, and the extent to which the product description matches the actual product. Electronic markets rely on reputation systems to ensure the quality of these characteristics for each merchant, and the importance of such systems is widely recognized in the literature (Resnick et al., 2000; Dellarocas, 2003). Typically, merchants' reputation in electronic markets is encoded by a "*reputation profile*" that includes: (a) the number of past transactions for the merchant, (b) a summary of numeric ratings from buyers who have completed transactions with the seller, and (c) a chronological list of textual feedback provided by these buyers.

Studies of online reputation, thus far, base a merchant's reputation on the *numeric* rating that characterizes the seller (e.g., average number of stars and number of completed transactions) (Melnik and Alm, 2002). The general conclusion of these studies show that merchants with higher (numeric) reputation can charge higher prices than the competition, for the same products, and still manage to make a sale. This *price premium* that the merchants can command over the competition is a measure of their reputation.

Definition 2.1 Consider a set of merchants $s_1, \ldots, s_n$ selling a product for prices $p_1, \ldots, p_n$. If s_i makes

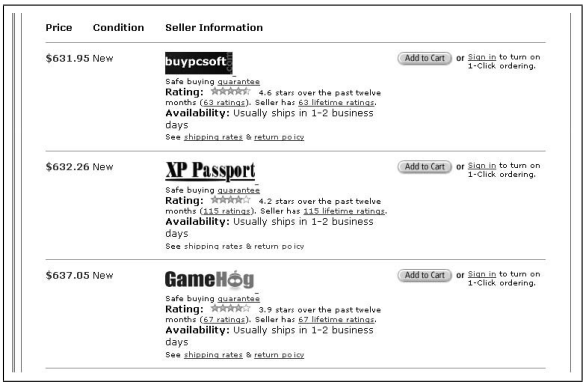


Figure 1: A set of merchants on Amazon.com selling an identical product for different prices

the sale for price p_i, then s_i commands a *price premium* equal to $p_i - p_j$ over s_j and a *relative price premium* equal to $\frac{p_i - p_j}{p_i}$. Hence, a transaction that involves n competing merchants generates $n-1$ price premiums.[1] The *average price premium* for the transaction is $\frac{\sum_{j \neq i}(p_i - p_j)}{n-1}$ and the *average relative price premium* is $\frac{\sum_{j \neq i}(p_i - p_j)}{p_i(n-1)}$. □

Example 2.1 Consider the case in Figure 1 where three merchants sell the same product for $631.95, $632.26, and $637.05, respectively. If *GameHog* sells the product, then the *price premium* against *XP Passport* is $4.79 (= $637.05 − $632.26) and against the merchant *BuyPCsoft* is $5.10. The *relative price premium* is 0.75% and 0.8%, respectively. Similarly, the *average price premium* for this transaction is $4.95 and the *average relative price premium* 0.78%. □

Different sellers in these markets derive their reputation from different characteristics: some sellers have a reputation for fast delivery, while some others have a reputation of having the lowest price among their peers. Similarly, while some sellers are praised for their packaging in the feedback, others get good comments for selling high-quality goods but are criticized for being rather slow with shipping. Even though previous studies have established the positive correlation between higher (numeric) reputation and higher price premiums, they ignored completely the role of the textual feedback and, in turn, the multi-dimensional nature of reputation in electronic markets. We show that the textual feedback adds significant *additional* value to the numerical scores, and affects the pricing power of the merchants.

[1] As an alternative definition we can ignore the *negative* price premiums. The experimental results are similar for both versions.

3 Data

We compiled a data set using software resellers from publicly available information on software product listings at Amazon.com. Our data set includes 280 individual software titles. The sellers' reputation matters when selling identical goods, and the price variation observed can be attributed primarily to variation in the merchant's reputation. We collected the data using Amazon Web Services over a period of 180 days, between October 2004 and March 2005. We describe below the two categories of data that we collected.

Transaction Data: The first part of our data set contains details of the transactions that took place on the marketplace of Amazon.com for each of the software titles. The Amazon Web Services associates a unique *transaction ID* for each unique product listed by a seller. This transaction ID enables us to distinguish between multiple or successive listings of identical products sold by the same merchant. Keeping with the methodology in prior research (Ghose et al., 2006), we crawl the Amazon's XML listings every 8 hours and when a transaction ID associated with a particular listing is removed, we infer that the listed product was successfully sold in the prior 8 hour window.[2] For each transaction that takes place, we keep the price at which the product was sold and the merchant's reputation at the time of the transaction (more on this later). Additionally, for each of the *competing listings for identical products*, we keep the listed price along with the competitors reputation. Using the collected data, we compute the price premium variables for each transaction[3] using Definition 2.1. Overall, our data set contains 1,078 merchants, 9,484 unique transactions and 107,922 price premiums (recall that each transaction generates multiple price premiums).

Reputation Data: The second part of our data set contains the reputation history of each merchant that had a (monitored) product for sale during our 180-day window. Each of these merchants has a feedback profile, which consists of numerical scores and text-based feedback, posted by buyers. We had an average of 4,932 postings per merchant. The numerical ratings

[2] Amazon indicates that their seller listings remain on the site indefinitely until they are sold and sellers can change the price of the product without altering the transaction ID.

[3] Ideally, we would also include the tax and shipping cost charged by each merchant in the computation of the price premiums. Unfortunately, we could not capture these costs using our methodology. Assuming that the fees for shipping and tax are independent of the merchants' reputation, our analysis is not affected.

are provided on a scale of one to five stars. These ratings are averaged to provide an overall score to the seller. Note that we collect all feedback (both numerical and textual) associated with a seller over the entire lifetime of the seller and we reconstruct each seller's exact feedback profile at the time of each transaction.

4 Econometrics-based Opinion Mining

In this section, we describe how we combine econometric techniques with NLP techniques to derive the semantic orientation and strength of the feedback evaluations. Section 4.1 describes how we structure the textual feedback and Section 4.2 shows how we use econometrics to estimate the polarity and strength of the evaluations.

4.1 Retrieving the Dimensions of Reputation

We characterize a merchant using a vector of reputation dimensions $X = (X_1, X_2, ..., X_n)$, representing its ability on each of n dimensions. We assume that each of these n dimensions is expressed by a *noun*, *noun phrase*, *verb*, or a *verb phrase* chosen from the set of all feedback postings, and that a merchant is evaluated on these n dimensions. For example, dimension 1 might be "*shipping*", dimension 2 might be "*packaging*" and so on. In our model, each of these dimensions is assigned a numerical score. Of course, when posting textual feedback, buyers do not assign explicit numeric scores to any dimension. Rather, they use *modifiers* (typically adjectives or adverbs) to evaluate the seller along each of these dimensions (we describe how we assign numeric scores to each modifier in Section 4.2). Once we have identified the set of all dimensions, we can then parse each of the feedback postings, associate a modifier with each dimension, and represent a feedback posting as an n-dimensional vector ϕ of modifiers.

Example 4.1 Suppose dimension 1 is "*delivery*," dimension 2 is "*packaging*," and dimension 3 is "*service*." The feedback posting *"I was impressed by the speedy delivery! Great service!"* is then encoded as $\phi_1 = [speedy, NULL, great]$, while the posting *"The item arrived in awful packaging, and the delivery was slow"* is encoded as $\phi_2 = [slow, awful, NULL]$. □

Let $\mathcal{M} = \{NULL, \mu_1, ..., \mu_M\}$ be the set of modifiers and consider a seller s_i with p postings in its reputation profile. We denote with $\mu^i_{jk} \in \mathcal{M}$ the modifier that appears in the j-th posting and is used to assess the k-th reputation dimension. We then structure the merchant's feedback as an $n \times p$ matrix $\mathbf{M}(s_i)$ whose rows are the p encoded vectors of modifiers associated with the seller. We construct $\mathbf{M}(s_i)$ as follows:

1. Retrieve the postings associated with a merchant.
2. Parse the postings to identify the dimensions across which the buyer evaluates a seller, keeping[4] the nouns, noun phrases, verbs, and verbal phrases as reputation characteristics.[5].
3. Retrieve adjectives and adverbs that refer to[6] dimensions (Step 2) and construct the ϕ vectors.

We have implemented this algorithm on the feedback postings of each of our sellers. Our analysis yields 151 unique dimensions, and a total of 142 modifiers (note that the same modifier can be used to evaluate multiple dimensions).

4.2 Scoring the Dimensions of Reputation

As discussed above, the textual feedback profile of merchant s_i is encoded as a $n \times p$ matrix $\mathbf{M}(s_i)$; the elements of this matrix belong to the set of modifiers $\mathcal{M}$. In our case, we are interested in computing the "score" $a(\mu, d, j)$ that a modifier $\mu \in \mathcal{M}$ assigns to the dimension d, when it appears in the j-th posting.

Since buyers tend to read only the first few pages of text-based feedback, we weight higher the influence of recent text postings. We model this by assuming that K is the number of postings that appear on each page ($K = 25$ on Amazon.com), and that c is the probability of clicking on the "Next" link and moving the next page of evaluations.[7] This assigns a posting-specific weight $r_j = c^{\lfloor \frac{j}{K} \rfloor} / \sum_{q=1}^{p} c^{\lfloor \frac{q}{K} \rfloor}$ for the j^{th} posting, where j is the rank of the posting, K is the number of postings per page, and p is the total number of postings for the given seller. Then, we set $a(\mu, d, j) = r_j \cdot a(\mu, d)$ where $a(\mu, d)$ is the "global" score that modifier μ assigns to dimension d.

Finally, since each reputation dimension has potentially a different weight, we use a weight vector $\mathbf{w}$ to

[4]We eliminate all dimensions appearing in the profiles of less than 50 (out of 1078) merchants, since we cannot extract statistically meaningful results for such sparse dimensions

[5]The technique as described in this paper, considers words like "shipping" and " delivery" as separate dimensions, although they refer to the same "real-life" dimension. We can use Latent Dirichlet Allocation (Blei et al., 2003) to reduce the number of dimensions, but this is outside the scope of this paper.

[6]To associate the adjectives and adverbs with the correct dimensions, we use the Collins HeadFinder capability of the Stanford NLP Parser.

[7]We report only results for $c = 0.5$. We conducted experiments other values of c as well and the results are similar.

weight the contribution of each reputation dimension to the overall "reputation score" $\Pi(s_i)$ of seller s_i:

$$\Pi(s_i) = \mathbf{r}^T \cdot \mathbf{A}(\mathbf{M}(s_i)) \cdot \mathbf{w} \quad (1)$$

where $\mathbf{r}^T = [r_1, r_2, ...r_p]$ is the vector of the posting-specific weights and $\mathbf{A}(\mathbf{M}(i))$ is a matrix that contains as element the score $a(\mu_j, d_k)$ where $\mathbf{M}(s_i)$ contains the modifier μ_j in the column of the dimension d_k. If we model the buyers' preferences as independently distributed along each dimension and each modifier score $a(\mu, d_k)$ also as an independent random variable, then the random variable $\Pi(s_i)$ is a sum of random variables. Specifically, we have:

$$\Pi(s_i) = \sum_{j=1}^{M} \sum_{k=1}^{n} \left(w_k \cdot a(\mu_j, d_k)\right) R(\mu_j, d_k) \quad (2)$$

where $R(\mu_j, d_k)$ is equal to the sum of the r_i weights across all postings in which the modifier μ_j modifies dimension d_k. We can easily compute the $R(\mu_j, d_k)$ values by simply counting appearances and weighting each appearance using the definition of r_i.

The question is, of course, how to estimate the values of $w_k \cdot a(\mu_j, d_k)$, which determine the polarity and intensity of the modifier μ_j modifying the dimension d_k. For this, we observe that the appearance of such modifier-dimension opinion phrases has an effect on the price premiums that a merchant can charge. Hence, there is a correlation between the reputation scores $\Pi(\cdot)$ of the merchants and the price premiums observed for each transaction. To discover the level of association, we use regression. Since we are dealing with panel data, we estimate *ordinary-least-squares (OLS) regression with fixed effects* (Greene, 2002), where the dependent variable is the *price premium* variable, and the independent variables are the reputation scores $\Pi(\cdot)$ of the merchants, together with a few other control variables. Generally, we estimate models of the form:

$$PricePremium_{ij} = \sum \beta_c \cdot X_{cij} + f_{ij} + \epsilon_{ij} + \beta_{t1} \cdot \Pi(merchant)_{ij} + \beta_{t2} \cdot \Pi(competitor)_{ij} \quad (3)$$

where $PricePremium_{ij}$ is one of the variations of price premium as given in Definition 2.1 for a seller s_i and product j, β_c, β_{t1}, and β_{t2} are the regressor coefficients, Xc are the control variables, $\Pi(\cdot)$ are the text reputation scores (see Equation 1), f_{ij} denotes the fixed effects and ϵ is the error term. In Section 5, we give the details about the control variables and the regression settings.

Interestingly, if we expand the $\Pi(\cdot)$ variables according to Equation 2, we can run the regression using the modifier-dimension pairs as independent variables, whose values are equal to the $R(\mu_j, d_k)$ values. After running the regression, the coefficients assigned to each modifier-dimension pair correspond to the value $w_k \cdot a(\mu_j, d_k)$ for each modifier-dimension pair. Therefore, we can easily estimate in economic terms the "value" of a particular modifier when used to evaluate a particular dimension.

5 Experimental Evaluation

In this section, we first present the experimental settings (Section 5.1), and then we describe the results of our experimental evaluation (Section 5.2).

5.1 Regression Settings

In Equation 3 we presented the general form of the regression for estimating the scores $a(\mu_j, d_k)$. Since we want to eliminate the effect of any other factors that may influence the price premiums, we also use a set of control variables. After all the control factors are taken into consideration, the modifier scores reflect the *additional* value of the text opinions. Specifically, we used as control variables the product's *price on Amazon*, the average *star rating of the merchant*, the *number of merchant's past transactions*, and the *number of sellers* for the product.

First, we ran OLS regressions with product-seller fixed effects controlling for unobserved heterogeneity across sellers and products. These fixed effects control for average product quality and differences in seller characteristics. We run multiple variations of our model, using different versions of the "price premium" variable as listed in Definition 2.1. We also tested variations where we include as independent variable not the individual reputation scores but the difference $\Pi(merchant) - \Pi(competitor)$. All regressions yielded qualitatively similar results, so due to space restrictions we only report results for the regressions that include all the control variables and all the text variables; we report results using the *price premium* as the dependent variable. Our regressions in this setting contain 107,922 observations, and a total of 547 independent variables.

5.2 Experimental Results

Recall of Extraction: The first step of our experimental evaluation is to examine whether the opinion extraction technique of Section 4.1 indeed captures all the reputation characteristics expressed in the feed-

Dimension	Human Recall	Computer Recall
Product Condition	0.76	0.76
Price	0.91	0.61
Package	0.96	0.66
Overall Experience	0.65	0.55
Delivery Speed	0.96	0.92
Item Description	0.22	0.43
Product Satisfaction	0.68	0.58
Problem Response	0.30	0.37
Customer Service	0.57	0.50
Average	**0.66**	**0.60**

Table 1: The recall of our technique compared to the recall of the human annotators

back (recall) and whether the dimensions that we capture are accurate (precision). To examine the recall question, we used two human annotators. The annotators read a random sample of 1,000 feedback postings, and identified the reputation dimensions mentioned in the text. Then, they examined the extracted modifier-dimension pairs for each posting and marked whether the modifier-dimension pairs captured the identified real reputation dimensions mentioned in the posting and which pairs were spurious, non-opinion phrases.

Both annotators identified nine reputation dimensions (see Table 1). Since the annotators did not agree in all annotations, we computed the average *human recall* $hRec_d = \frac{agreed_d}{all_d}$ for each dimension d, where $agreed_d$ is the number of postings for which both annotators identified the reputation dimension d, and all_d is the number of postings in which at least one annotator identified the dimension d. Based on the annotations, we computed the recall of our algorithm against each annotator. We report the average recall for each dimension, together with the human recall in Table 1. The recall of our technique is only slightly inferior to the performance of humans, indicating that the technique of Section 4.1 extracts the majority of the posted evaluations.[8]

Interestingly, precision is not an issue in our setting. In our framework, if an particular modifier-dimension pair is just noise, then it is almost impossible to have a statistically significant correlation with the price premiums. The noisy opinion phrases are statistically guaranteed to be filtered out by the regression.

Estimating Polarity and Strength: In Table 2, we present the modifier-dimension pairs (positive and negative) that had the strongest "dollar value" and were statistically significant across all regressions. (Due to space issues, we cannot list the values for all pairs.) These values reflect changes in the merchants's pricing power *after* taking their average numerical score and level of experience into account, and also highlight the *additional* the value contained in text-based reputation. The examples that we list here illustrate that our technique generates a natural ranking of the opinion phrases, inferring the strength of each modifier *within* the context in which this opinion is evaluated. This holds true even for misspelled evaluations that would break existing techniques based on annotation or on resources like WordNet. Furthermore, these values reflect the *context* in which the opinion is evaluated. For example, the pair *good packaging* has a dollar value of -$0.58. Even though this seems counterintuitive, it actually reflects the nature of an online marketplace where most of the positive evaluations contain superlatives, and a mere "good" is actually interpreted by the buyers as a lukewarm, slightly negative evaluation. Existing techniques cannot capture such phenomena.

Price Premiums vs. Ratings: One of the natural comparisons is to examine whether we could reach similar results by just using the average star rating associated with each feedback posting to infer the score of each opinion phrase. The underlying assumption behind using the ratings is that the review is perfectly summarized by the star rating, and hence the text plays mainly an explanatory role and carries no extra information, given the star rating. For this, we examined the R^2 fit of the regression, with and without the use of the text variables. Without the use of text variables, the R^2 was 0.35, while when using only the text-based regressors, the R^2 fit increased to 0.63. This result clearly indicates that the actual text contains significantly more information than the ratings.

We also experimented with predicting which merchant will make a sale, if they simultaneously sell the same product, based on their listed prices and on their numeric and text reputation. Our C4.5 classifier (Quinlan, 1992) takes a pair of merchants and decides which of the two will make a sale. We used as training set the transactions that took place in the first four months and as test set the transactions in the last two months of our data set. Table 3 summarizes the results for different sets of features used. The 55%

[8] In the case of "Item Description," where the computer recall was higher than the human recall, our technique identified almost all the phrases of one annotator, but the other annotator had a more liberal interpretation of "Item Description" dimension and annotated significantly more postings with the dimension "Item Description" than the other annotator, thus decreasing the human recall.

Modifier Dimension	Dollar Value
[wonderful experience]	$5.86
[outstanding seller]	$5.76
[excellant service]	$5.27
[lightning delivery]	$4.84
[highly recommended]	$4.15
[best seller]	$3.80
[perfectly packaged]	$3.74
[excellent condition]	$3.53
[excellent purchase]	$3.22
[excellent seller]	$2.70
[excellent communication]	$2.38
[perfect item]	$1.92
[terrific condition]	$1.87
[top quality]	$1.67
[awesome service]	$1.05
[A+++ seller]	$1.03
[great merchant]	$0.93
[friendly service]	$0.81
[easy service]	$0.78
[never received]	-$7.56
[defective product]	-$6.82
[horible experience]	-$6.79
[never sent]	-$6.69
[never recieved]	-$5.29
[bad experience]	-$5.26
[cancelled order]	-$5.01
[never responded]	-$4.87
[wrong product]	-$4.39
[not as advertised]	-$3.93
[poor packaging]	-$2.92
[late shipping]	-$2.89
[wrong item]	-$2.50
[not yet received]	-$2.35
[still waiting]	-$2.25
[wrong address]	-$1.54
[never buy]	-$1.48

Table 2: The highest scoring opinion phrases, as determined by the product $w_k \cdot a(\mu_j, d_k)$.

accuracy when using only prices as features indicates that customers rarely choose a product based solely on price. Rather, as indicated by the 74% accuracy, they also consider the reputation of the merchants. However, the real value of the postings relies on the text and not on the numeric ratings: the accuracy is 87%-89% when using the textual reputation variables. In fact, text subsumes the numeric variables but not vice versa, as indicated by the results in Table 3.

6 Related Work

To the best of our knowledge, our work is the first to use economics for measuring the effect of opinions and deriving their polarity and strength in an econometric manner. A few papers in the past tried to combine text analysis with economics (Das and Chen, 2006; Lewitt and Syverson, 2005), but the text analysis was limited to token counting and did not use any NLP techniques. The technique of Section 4.1 is based on existing research in sentiment analysis. For instance, (Hatzivassiloglou and McKeown, 1997; Nigam and Hurst, 2004) use annotated data to create a supervised learning technique to identify the semantic orientation of adjectives. We follow the approach by Turney (2002), who note that the semantic orientation of an adjective depends on the noun that it modifies and suggest using adjective-noun or adverb-verb pairs to extract semantic orientation. However, we do not rely on linguistic resources (Kamps and Marx, 2002) or on search engines (Turney and Littman, 2003) to determine the semantic orientation, but rather rely on econometrics for this task. Hu and Liu (2004), whose study is the closest to our work, use WordNet to compute the semantic orientation of product evaluations and try to summarize user reviews by extracting the positive and negative evaluations of the different product features. Similarly, Snyder and Barzilay (2007) decompose an opinion across several dimensions and capture the sentiment across each dimension. Other work in this area includes (Lee, 2004; Popescu and Etzioni, 2005) which uses text mining in the context product reviews, but none uses the economic context to evaluate the opinions.

Features	Accuracy on Test Set
Price	55%
Price + Numeric Reputation	74%
Price + Numeric Reputation + Text Reputation	89%
Price + Text Reputation	87%

Table 3: Predicting the merchant who makes the sale.

7 Conclusion and Further Applications

We demonstrated the value of using econometrics for extracting a *quantitative* interpretation of opinions. Our technique, additionally, takes into consideration the *context* within which these opinions are evaluated. Our experimental results show that our techniques can capture the pragmatic meaning of the expressed opinions using simple economic variables as a form of training data. The source code with our implementation together with the data set used in this paper are available from `http://economining.stern.nyu.edu`.

There are many other applications beyond reputation systems. For example, using sales rank data from Amazon.com, we can examine the effect of product reviews on product sales and detect the weight that

customers put on different product features; furthermore, we can discover how customer evaluations on individual product features affect product sales and extract the pragmatic meaning of these evaluations. Another application is the analysis of the effect of news stories on stock prices: we can examine what news topics are important for the stock market and see how the views of different opinion holders and the wording that they use can cause the market to move up or down. In a slightly different twist, we can analyze news stories and blogs in conjunction with results from prediction markets and extract the pragmatic effect of news and blogs on elections or other political events. Another research direction is to examine the effect of summarizing product descriptions on product sales: short descriptions reduce the cognitive load of consumers but increase their uncertainty about the underlying product characteristics; a longer description has the opposite effect. The optimum description length is the one that balances both effects and maximizes product sales.

Similar approaches can improve the state of art in both economics and computational linguistics. In economics and in social sciences in general, most researchers handle textual data manually or with simplistic token counting techniques; in the worst case they ignore text data altogether. In computational linguistics, researchers often rely on human annotators to generate training data, a laborious and error-prone task. We believe that cross-fertilization of ideas between the fields of computational linguistics and econometrics can be beneficial for both fields.

Acknowledgments

The authors would like to thank Elena Filatova for the useful discussions and the pointers to related literature. We also thank Sanjeev Dewan, Alok Gupta, Bin Gu, and seminar participants at Carnegie Mellon University, Columbia University, Microsoft Research, New York University, Polytechnic University, and University of Florida for their comments and feedback. We thank Rhong Zheng for assistance in data collection. This work was partially supported by a Microsoft Live Labs Search Award, a Microsoft Virtual Earth Award, and by NSF grants IIS-0643847 and IIS-0643846. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the Microsoft Corporation or of the National Science Foundation.

References

D.M. Blei, A.Y. Ng, and M.I. Jordan. 2003. Latent Dirichlet allocation. *JMLR*, 3:993–1022.

E. Breck, Y. Choi, and C. Cardie. 2007. Identifying expressions of opinion in context. In *IJCAI-07*, pages 2683–2688.

H. Cui, V. Mittal, and M. Datar. 2006. Comparative experiments on sentiment classification for online product reviews. In *AAAI-2006*.

S. Ranjan Das and M. Chen. 2006. Yahoo! for Amazon: Sentiment extraction from small talk on the web. Working Paper, Santa Clara University.

K. Dave, S. Lawrence, and D.M. Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *WWW12*, pages 519–528.

C. Dellarocas. 2003. The digitization of word-of-mouth: Promise and challenges of online reputation mechanisms. *Management Science*, 49(10):1407–1424.

A. Ghose, M.D. Smith, and R. Telang. 2006. Internet exchanges for used books: An empirical analysis for product cannibalization and social welfare. *Information Systems Research*, 17(1):3–19.

W.H. Greene. 2002. *Econometric Analysis*. 5th edition.

V. Hatzivassiloglou and K.R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *ACL'97*, pages 174–181.

M. Hu and B. Liu. 2004. Mining and summarizing customer reviews. In *KDD-2004*, pages 168–177.

J. Kamps and M. Marx. 2002. Words with attitude. In *Proceedings of the First International Conference on Global WordNet*.

S.-M. Kim and E. Hovy. 2004. Determining the sentiment of opinions. In *COLING 2004*, pages 1367–1373.

A.C. König and E. Brill. 2006. Reducing the human overhead in text categorization. In *KDD-2006*, pages 598–603.

T. Lee. 2004. Use-centric mining of customer reviews. In *WITS*.

S. Lewitt and C. Syverson. 2005. Market distortions when agents are better informed: The value of information in real estate transactions. Working Paper, University of Chicago.

M.I. Melnik and J. Alm. 2002. Does a seller's reputation matter? Evidence from eBay auctions. *Journal of Industrial Economics*, 50(3):337–350, September.

K. Nigam and M. Hurst. 2004. Towards a robust metric of opinion. In *AAAI Spring Symposium on Exploring Attitude and Affect in Text*, pages 598–603.

B. Pang and L. Lee. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *EMNLP 2002*.

B. Pang and L. Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL 2004*, pages 271–278.

B. Pang and L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL 2005*.

A.-M. Popescu and O. Etzioni. 2005. Extracting product features and opinions from reviews. In *HLT/EMNLP 2005*.

B. Snyder and R. Barzilay. 2007. Multiple aspect ranking using the good grief algorithm. In *HLT-NAACL 2007*.

J.R. Quinlan. 1992. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, Inc.

P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. 2000. Reputation systems. *CACM*, 43(12):45–48, December.

P.D. Turney and M.L. Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems*, 21(4):315–346.

P.D. Turney. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *ACL 2002*, pages 417–424.

T. Wilson, J. Wiebe, and R. Hwa. 2006. Recognizing strong and weak opinion clauses. *Computational Intell.*, 22(2):73–99.

PageRanking WordNet Synsets: An Application to Opinion Mining*

Andrea Esuli and Fabrizio Sebastiani
Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
Via Giuseppe Moruzzi, 1 – 56124 Pisa, Italy
{andrea.esuli,fabrizio.sebastiani}@isti.cnr.it

Abstract

This paper presents an application of PageRank, a random-walk model originally devised for ranking Web search results, to ranking WordNet synsets in terms of how strongly they possess a given semantic property. The semantic properties we use for exemplifying the approach are positivity and negativity, two properties of central importance in sentiment analysis. The idea derives from the observation that WordNet may be seen as a graph in which synsets are connected through the binary relation "a term belonging to synset s_k occurs in the gloss of synset s_i", and on the hypothesis that this relation may be viewed as a transmitter of such semantic properties. The data for this relation can be obtained from eXtended WordNet, a publicly available sense-disambiguated version of WordNet. We argue that this relation is structurally akin to the relation between hyperlinked Web pages, and thus lends itself to PageRank analysis. We report experimental results supporting our intuitions.

1 Introduction

Recent years have witnessed an explosion of work on *opinion mining* (aka *sentiment analysis*), the discipline that deals with the quantitative and qualitative analysis of text for the purpose of determining its opinion-related properties (ORPs). An important part of this research has been the work on the automatic determination of the ORPs of *terms*, as e.g., in determining whether an adjective tends to give a positive, a negative, or a neutral nature to the noun phrase it appears in. While many works (Esuli and Sebastiani, 2005; Hatzivassiloglou and McKeown, 1997; Kamps et al., 2004; Takamura et al., 2005; Turney and Littman, 2003) view the properties of positivity and negativity as categorical (i.e., a term is either positive or it is not), others (Andreevskaia and Bergler, 2006b; Grefenstette et al., 2006; Kim and Hovy, 2004; Subasic and Huettner, 2001) view them as *graded* (i.e., a term may be positive to a certain degree), with the underlying interpretation varying from fuzzy to probabilistic.

Some authors go a step further and attach these properties not to terms but to term *senses* (typically: WordNet synsets), on the assumption that different senses of the same term may have different opinion-related properties (Andreevskaia and Bergler, 2006a; Esuli and Sebastiani, 2006b; Ide, 2006; Wiebe and Mihalcea, 2006).

In this paper we contribute to this latter literature with a novel method for ranking the *entire* set of WordNet synsets, irrespectively of POS, according to their ORPs. Two rankings are produced, one according to positivity and one according to negativity. The two rankings are independent, i.e., it is *not* the case that one is the inverse of the other, since e.g., the least positive synsets may be negative or neutral synsets alike.

This work was partially supported by Project ONTOTEXT "From Text to Knowledge for the Semantic Web", funded by the Provincia Autonoma di Trento under the 2004–2006 "Fondo Unico per la Ricerca" funding scheme.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 424–431,
Prague, Czech Republic, June 2007.

The main hypothesis underlying our method is that the positivity and negativity of WordNet synsets can be determined by mining their glosses. It crucially relies on the observation that the gloss of a WordNet synset contains terms *that themselves belong to synsets*, and on the hypothesis that the glosses of positive (resp. negative) synsets will mostly contain terms belonging to positive (negative) synsets. This means that the binary relation $s_i \blacktriangleright s_k$ ("the gloss of synset s_i contains a term belonging to synset s_k"), which induces a directed graph on the set of WordNet synsets, may be thought of as a channel through which positivity and negativity flow, from the *definiendum* (the synset s_i being defined) to the *definiens* (a synset s_k that contributes to the definition of s_i by virtue of its member terms occurring in the gloss of s_i). In other words, if a synset s_i is known to be positive (negative), this can be viewed as an indication that the synsets s_k to which the terms occurring in the gloss of s_i belong, are themselves positive (negative).

We obtain the data of the $\blacktriangleright$ relation from eXtended WordNet (Harabagiu et al., 1999), an automatically sense-disambiguated version of WordNet in which every term occurrence in every gloss is linked to the synset it is deemed to belong to.

In order to compute how polarity flows in the graph of WordNet synsets we use the well known PageRank algorithm (Brin and Page, 1998). PageRank, a random-walk model for ranking Web search results which lies at the basis of the Google search engine, is probably the most important single contribution to the fields of information retrieval and Web search of the last ten years, and was originally devised in order to detect how authoritativeness flows in the Web graph and how it is conferred onto Web sites. The advantages of PageRank are its strong theoretical foundations, its fast convergence properties, and the effectiveness of its results. The reason why PageRank, among all random-walk algorithms, is particularly suited to our application will be discussed in the rest of the paper.

Note however that our method is not limited to ranking synsets *by positivity and negativity*, and could in principle be applied to the determination of other semantic properties of synsets, such as membership in a domain, since for many other properties we may hypothesize the existence of a similar "hydraulics" between synsets. We thus see positivity and negativity only as proofs-of-concept for the potential of the method.

The rest of the paper is organized as follows. Section 2 reports on related work on the ORPs of lexical items, highlighting the similarities and differences between the discussed methods and our own. In Section 3 we turn to discussing our method; in order to make the paper self-contained, we start with a brief introduction of PageRank (Section 3.1) and of the structure of eXtended WordNet (Section 3.2). Section 4 describes the structure of our experiments, while Section 5 discusses the results we have obtained, comparing them with other results from the literature. Section 6 concludes.

2 Related work

Several works have recently tackled the automated determination of term polarity. Hatzivassiloglou and McKeown (1997) determine the polarity of adjectives by mining pairs of conjoined adjectives from text, and observing that conjunctions such as and tend to conjoin adjectives of the same polarity while conjunctions such as but tend to conjoin adjectives of opposite polarity. Turney and Littman (2003) determine the polarity of generic terms by computing the pointwise mutual information (PMI) between the target term and each of a set of "seed" terms of known positivity or negativity, where the marginal and joint probabilities needed for PMI computation are equated to the fractions of documents from a given corpus that contain the terms, individually or jointly. Kamps et al. (2004) determine the polarity of adjectives by checking whether the target adjective is closer to the term good or to the term bad in the graph induced on WordNet by the synonymy relation. Kim and Hovy (2004) determine the polarity of generic terms by means of two alternative learning-free methods that use two sets of seed terms of known positivity and negativity, and are based on the frequency with which synonyms of the target term also appear in the respective seed sets. Among these works, (Turney and Littman, 2003) has proven by far the most effective, but it is also by far the most computationally intensive.

Some recent works have employed, as in the present paper, the glosses from online dictionar-

ies for term polarity detection. Andreevskaia and Berger (2006a) extend a set of terms of known positivity/negativity by adding to them all the terms whose glosses contain them; this algorithm does not view glosses as a source for a graph of terms, and is based on a different intuition than ours. Esuli and Sebastiani (2005; 2006a) determine the ORPs of generic terms by learning, in a semi-supervised way, a binary term classifier from a set of training terms that have been given vectorial representations by indexing their WordNet glosses. The same authors later extend their work to determining the ORPs of WordNet synsets (Esuli and Sebastiani, 2006b). However, there is a substantial difference between these works and the present one, in that the former simply view the glosses as sources of textual representations for the terms/synsets, and not as inducing a graph of synsets as we instead view them here.

The work closest in spirit to the present one is probably that by Takamura et al. (2005), who determine the polarity of terms by applying intuitions from the theory of electron spins: two terms that appear one in the gloss of the other are viewed as akin to two neighbouring electrons, which tend to acquire the same "spin" (a notion viewed as akin to polarity) due to their being neighbours. This work is similar to ours since a graph between terms is generated from dictionary glosses, and since an iterative algorithm that converges to a stable state is used, but the algorithm is very different, and based on intuitions from very different walks of life.

Some recent works have tackled the attribution of opinion-related properties to word senses or synsets (Ide, 2006; Wiebe and Mihalcea, 2006)[1]; however, they do not use glosses in any significant way, and are thus very different from our method.

The interested reader may also consult (Mihalcea, 2006) for other applications of random-walk models to computational linguistics.

3 Ranking WordNet synsets by PageRank

3.1 The PageRank algorithm

Let $G = \langle N, L \rangle$ be a directed graph, with N its set of nodes and L its set of directed links; let $\mathbf{W_0}$ be the $|N| \times |N|$ *adjacency matrix* of G, i.e., the matrix such that $\mathbf{W}_0[i,j] = 1$ iff there is a link from node n_i to node n_j. We will denote by $B(i) = \{n_j \mid \mathbf{W_0}[j,i] = 1\}$ the set of the *backward neighbours* of n_i, and by $F(i) = \{n_j \mid \mathbf{W_0}[i,j] = 1\}$ the set of the *forward neighbours* of n_i. Let $\mathbf{W}$ be the *row-normalized adjacency matrix* of G, i.e., the matrix such that $\mathbf{W}[i,j] = \frac{1}{|F(i)|}$ iff $\mathbf{W_0}[i,j] = 1$ and $\mathbf{W}[i,j] = 0$ otherwise.

The input to PageRank is the row-normalized adjacency matrix $\mathbf{W}$, and its output is a vector $\mathbf{a} = \langle a_1, \ldots, a_{|N|} \rangle$, where a_i represents the "score" of node n_i. When using PageRank for search results ranking, n_i is a Web site and a_i measures its computed authoritativeness; in our application n_i is instead a synset and a_i measures the degree to which n_i has the semantic property of interest. PageRank iteratively computes vector $\mathbf{a}$ based on the formula

$$a_i^{(k)} \leftarrow \alpha \sum_{j \in B(i)} \frac{a_j^{(k-1)}}{|F(j)|} + (1-\alpha)e_i \qquad (1)$$

where $a_i^{(k)}$ denotes the value of the i-th entry of vector $\mathbf{a}$ at the k-th iteration, e_i is a constant such that $\sum_i e_{i=1}^{|N|} = 1$, and $0 \leq \alpha \leq 1$ is a control parameter. In vectorial form, Equation 1 can be written as

$$\mathbf{a}^{(k)} = \alpha \mathbf{a}^{(k-1)} \mathbf{W} + (1-\alpha)\mathbf{e} \qquad (2)$$

The underlying intuition is that a node n_i has a high score when (recursively) it has many high-scoring backward neighbours with few forward neighbours each; a node n_j thus passes its score a_j along to its forward neighbours $F(j)$, but this score is subdivided equally among the members of $F(j)$. This mechanism (that is represented by the summation in Equation 1) is then "smoothed" by the e_i constants, whose role is (see (Bianchini et al., 2005) for details) to avoid that scores flow and get trapped into so-called "rank sinks" (i.e., cliques with backward neighbours but no forward neighbours).

The computational properties of the PageRank algorithm, and how to compute it efficiently, have been widely studied; the interested reader may consult (Bianchini et al., 2005).

In the original application of PageRank for ranking Web search results the elements of $\mathbf{e}$ are usually taken to be all equal to $\frac{1}{|N|}$. However, it is possible

[1] Andreevskaia and Berger (2006a) also work on term senses, rather than terms, but they evaluate their work on terms only. This is the reason why they are listed in the preceding paragraph and not here.

to give different values to different elements in $\mathbf{e}$. In fact, the value of e_i amounts to an *internal source of score* for n_i that is constant across the iterations and independent from its backward neighbours. For instance, attributing a null e_i value to all but a few Web pages that are about a given topic can be used in order to bias the ranking of Web pages in favour of this topic (Haveliwala, 2003).

In this work we use the e_i values as internal sources of a given ORP (positivity *or* negativity), by attributing a null e_i value to all but a few "seed" synsets known to possess that ORP. PageRank will thus make the ORP flow from the seed synsets, at a rate constant throughout the iterations, into other synsets along the ▶ relation, until a stable state is reached; the final a_i values can be used to rank the synsets in terms of that ORP. Our method thus requires *two* runs of PageRank; in the first $\mathbf{e}$ has non-null scores for the positive seed synsets, while in the second the same happens for the negative ones.

3.2 eXtended WordNet

The transformation of WordNet into a graph based on the ▶ relation would of course be non-trivial, but is luckily provided by *eXtended WordNet* (Harabagiu et al., 1999), a publicly available version of WordNet in which (among other things) each term s_k occurring in a WordNet gloss (except those in example phrases) is lemmatized and mapped to the synset in which it belongs[2]. We use eXtended WordNet version 2.0-1.1, which refers to WordNet version 2.0. The eXtended WordNet resource has been automatically generated, which means that the associations between terms and synsets are likely to be sometimes incorrect, and this of course introduces noise in our method.

3.3 PageRank, (eXtended) WordNet, and ORP flow

We now discuss the application of PageRank to ranking WordNet synsets by positivity and negativity. Our algorithm consists in the following steps:

1. The graph $G = \langle N, L \rangle$ on which PageRank will be applied is generated. We define N to be the set of all WordNet synsets; in WordNet 2.0 there are 115,424 of them. We define L to contain a link from synset s_i to synset s_k iff the gloss of s_i contains *at least* a term belonging to s_k (terms occurring in the examples phrases and terms occurring after a term that expresses negation are not considered). Numbers, articles and prepositions occurring in the glosses are discarded, since they can be assumed to carry no positivity and negativity, and since they do not belong to a synset of their own. This leaves only nouns, adjectives, verbs, and adverbs.

2. The graph $G = \langle N, L \rangle$ is "pruned" by removing "self-loops", i.e., links going from a synset s_i into itself (since we assume that there is no flow of semantics from a concept unto itself). The row-normalized adjacency matrix $\mathbf{W}$ of G is derived.

3. The e_i values are loaded into the $\mathbf{e}$ vector; all synsets other than the seed synsets of renowned positivity (negativity) are given a value of 0. The α control parameter is set to a fixed value. We experiment with several different versions of the $\mathbf{e}$ vector and several different values of α; see Section 4.3 for details.

4. PageRank is executed using $\mathbf{W}$ and $\mathbf{e}$, iterating until a predefined termination condition is reached. The termination condition we use in this work consists in the fact that the cosine of the angle between $\mathbf{a}^{(k)}$ and $\mathbf{a}^{(k+1)}$ is above a predefined threshold χ (here we have set $\chi = 1 - 10^{-9}$).

5. We rank all the synsets of WordNet in descending order of their a_i score.

The process is run twice, once for positivity and once for negativity.

The last question to be answered is: "why PageRank?" Are the characteristics of PageRank more suitable to the problem of ranking synsets than other random-walk algorithms? The answer is yes, since it seems reasonable that:

1. If terms contained in synset s_k occur in the glosses of many positive synsets, and if the positivity scores of these synsets are high, then it is likely that s_k is itself positive (the same happens for negativity). This justifies the summation of Equation 1.

[2] http://xwn.hlt.utdallas.edu/

2. If the gloss of a positive synset that contains a term in synset s_k also contains many other terms, then this is a weaker indication that s_k is itself positive (this justifies dividing by $|F(j)|$ in Equation 1).

3. The ranking resulting from the algorithm needs to be biased in favour of a specific ORP; this justifies the presence of the $(1-\alpha)e_i$ factor in Equation 1).

The fact that PageRank is the "right" random-walk algorithm for our application is also confirmed by some experiments (not reported here for reasons of space) we have run with slightly different variants of the model (e.g., one in which we challenge intuition 2 above and thus avoid dividing by $|F(j)|$ in Equation 1). These experiments have always returned inferior results with respect to standard PageRank, thereby confirming the correctness of our intuitions.

4 Experiments

4.1 The benchmark

To evaluate the quality of the rankings produced by our experiments we have used the Micro-WNOp corpus (Cerini et al., 2007) as a benchmark[3]. Micro-WNOp consists in a set of 1,105 WordNet synsets, each of which was manually assigned a triplet of scores, one of positivity, one of negativity, one of neutrality. The evaluation was performed by five MSc students of linguistics, proficient second-language speakers of English. Micro-WNOp is representative of WordNet with respect to the different parts of speech, in the sense that it contains synsets of the different parts of speech in the same proportions as in the entire WordNet. However, it is not representative of WordNet with respect to ORPs, since this would have brought about a corpus largely composed of neutral synsets, which would be pretty useless as a benchmark for testing automatically derived lexical resources *for opinion mining*. It was thus generated by randomly selecting 100 positive + 100 negative + 100 neutral terms from the General Inquirer lexicon (see (Turney and Littman, 2003) for details) and including all the synsets that contained at least one such term, without paying attention to POS. See (Cerini et al., 2007) for more details.

The corpus is divided into three parts:

- Common: 110 synsets which all the evaluators evaluated by working together, so as to align their evaluation criteria.
- Group1: 496 synsets which were each independently evaluated by three evaluators.
- Group2: 499 synsets which were each independently evaluated by the other two evaluators.

Each of these three parts has the same balance, in terms of both parts of speech and ORPs, of Micro-WNOp as a whole. We obtain the positivity (negativity) ranking from Micro-WNOp by averaging the positivity (negativity) scores assigned by the evaluators of each group into a single score, and by sorting the synsets according to the resulting score. We use Group1 as a validation set, i.e., in order to fine-tune our method, and Group2 as a test set, i.e., in order to evaluate our method once all the parameters have been optimized on the validation set.

The result of applying PageRank to the graph G induced by the $\blacktriangleright$ relation, given a vector $\mathbf{e}$ of internal sources of positivity (negativity) score and a value for the α parameter, is a ranking of all the WordNet synsets in terms of positivity (negativity). By using different $\mathbf{e}$ vectors and different values of α we obtain different rankings, whose quality we evaluate by comparing them against the ranking obtained from Micro-WNOp.

4.2 The effectiveness measure

A ranking $\preceq$ is a partial order on a set of objects $N = \{o_1 \dots o_{|N|}\}$. Given a pair (o_i, o_j) of objects, o_i may precede o_j ($o_i \preceq o_j$), it may follow o_i ($o_i \succeq o_j$), or it may be tied with o_j ($o_i \approx o_j$).

To evaluate the rankings produced by PageRank we have used the *p-normalized Kendall τ distance* (noted τ_p – see e.g., (Fagin et al., 2004)) between the Micro-WNOp rankings and those predicted by PageRank. A standard function for the evaluation of rankings with ties, τ_p is defined as

$$\tau_p = \frac{n_d + p \cdot n_u}{Z} \quad (3)$$

[3] http://www.unipv.it/wnop/

where n_d is the number of *discordant pairs*, i.e., pairs of objects ordered one way in the gold standard and the other way in the prediction; n_u is the number of pairs ordered (i.e., not tied) in the gold standard and tied in the prediction, and p is a penalization to be attributed to each such pair; and Z is a normalization factor (equal to the number of pairs that are ordered in the gold standard) whose aim is to make the range of τ_p coincide with the $[0, 1]$ interval. Note that pairs tied in the gold standard are not considered in the evaluation.

The penalization factor is set to $p = \frac{1}{2}$, which is equal to the probability that a ranking algorithm correctly orders the pair by random guessing; there is thus no advantage to be gained from either random guessing or assigning ties between objects. For a prediction which perfectly coincides with the gold standard τ_p equals 0; for a prediction which is exactly the inverse of the gold standard τ_p equals 1.

4.3 Setup

In order to produce a ranking by positivity (negativity) we need to provide an $\mathbf{e}$ vector as input to PageRank. We have experimented with several different definitions of $\mathbf{e}$, each for both positivity and negativity. For reasons of space, we only report results from the five most significant ones.

We have first tested a vector (hereafter dubbed $\mathbf{e}1$) with all values uniformly set to $\frac{1}{|N|}$. This is the $\mathbf{e}$ vector originally used in (Brin and Page, 1998) for Web page ranking, and brings about an unbiased (that is, with respect to particular properties) ranking of WordNet. Of course, it is not meant to be used for ranking by positivity or negativity; we have used it as a baseline in order to evaluate the impact of property-biased vectors.

The first sensible, albeit minimalistic, definition of $\mathbf{e}$ we have used (dubbed $\mathbf{e}2$) is that of a vector with uniform non-null e_i scores assigned to the synsets that contain the adjective good (bad), and null scores for all other synsets. A further, still fairly minimalistic definition we have used (dubbed $\mathbf{e}3$) is that of a vector with uniform non-null e_i scores assigned to the synsets that contain at least one of the seven "paradigmatic" positive (negative) adjectives used as seeds in (Turney and Littman, 2003)[4], and null scores for all other synsets.

We have also tested a more complex version of $\mathbf{e}$, with e_i scores obtained from release 1.0 of SentiWordNet (Esuli and Sebastiani, 2006b)[5]. This latter is a lexical resource in which each WordNet synset is given a positivity score, a negativity score, and a neutrality score. We produced an $\mathbf{e}$ vector (dubbed $\mathbf{e}4$) in which the score assigned to a synset is proportional to the positivity (negativity) score assigned to it by SentiWordNet, and in which all entries sum up to 1. In a similar way we also produced a further $\mathbf{e}$ vector (dubbed $\mathbf{e}5$) through the scores of a newer release of SentiWordNet (release 1.1), resulting from a slight modification of the approach that had brought about release 1.0 (Esuli and Sebastiani, 2007b).

PageRank is parametric on α, which determines the balance between the contributions of the $\mathbf{a}^{(k-1)}$ vector and the $\mathbf{e}$ vector. A value of $\alpha = 0$ makes the $\mathbf{a}^{(k)}$ vector coincide with $\mathbf{e}$, and corresponds to discarding the contribution of the random-walk algorithm. Conversely, setting $\alpha = 1$ corresponds to discarding the contribution of $\mathbf{e}$, and makes $\mathbf{a}^{(k)}$ uniquely depend on the topology of the graph; the result is an "unbiased" ranking. The desirable cases are, of course, in between. As first hinted in Section 4.1, we thus optimize the α parameter on the synsets in Group1, and then test the algorithm with the optimal value of α on the synsets in Group2. All the 101 values of α from 0.0 to 1.0 with a step of .01 have been tested in the optimization phase. Optimization is performed anew for each experiment, which means that different values of α may be eventually selected for different $\mathbf{e}$ vectors.

5 Results

The results show that the use of PageRank in combination with suitable vectors $\mathbf{e}$ almost always improves the ranking, sometimes significantly so, with respect to the original ranking embodied by the $\mathbf{e}$ vector.

For positivity, the rankings produced using PageRank and any of the vectors from $\mathbf{e}2$ to $\mathbf{e}5$ all improve on the original rankings, with a relative improvement, measured as the relative decrease in τ_p,

[4]The seven positive adjectives are good, nice, excellent, positive, fortunate, correct, superior, and the seven negative ones are bad, nasty, poor, negative, unfortunate, wrong, inferior.

[5]http://sentiwordnet.isti.cnr.it/

ranging from -4.88% ($\mathbf{e}5$) to -6.75% ($\mathbf{e}4$). These rankings are also all better than the rankings produced by using PageRank and the uniform-valued vector $\mathbf{e}1$, with a minimum relative improvement of -5.04% ($\mathbf{e}3$) and a maximum of -34.47% ($\mathbf{e}4$). This suggests that the key to good performance is indeed a combination of positivity flow *and* internal source of score.

For the negativity rankings, the performance of both SentiWordNet-based vectors is still good, producing a -4.31% ($\mathbf{e}4$) and a -3.45% ($\mathbf{e}5$) improvement with respect to the original rankings. The "minimalistic" vectors (i.e., $\mathbf{e}2$ and $\mathbf{e}3$) are not as good as their positive counterparts. The reason seems to be that the generation of a ranking by negativity seems a somehow harder task than the generation of a ranking by positivity; this is also shown by the results obtained with the uniform-valued vector $\mathbf{e}1$, in which the application of PageRank improves with respect to $\mathbf{e}1$ for positivity but deteriorates for negativity. However, against the baseline constituted by the results obtained with the uniform-valued vector $\mathbf{e}1$ for negativity, our rankings show a relevant improvement, ranging from -8.56% ($\mathbf{e}2$) to -48.27% ($\mathbf{e}4$).

Our results are particularly significant for the $\mathbf{e}4$ vectors, derived by SentiWordNet 1.0, for a number of reasons. First, $\mathbf{e}4$ brings about the best value of τ_p obtained in all our experiments (.325 for positivity, .284 for negativity). Second, the relative improvement with respect to $\mathbf{e}4$ is the most marked among the various choices for $\mathbf{e}$ (6.75% for positivity, 4.31% for negativity). Third, the improvement is obtained with respect to an already high-quality resource, obtained by the same techniques that, at the term level, are still the best performers for polarity detection on the widely used General Inquirer benchmark (Esuli and Sebastiani, 2005).

Finally, observe that the fact that $\mathbf{e}4$ outperforms all other choices for $\mathbf{e}$ (and $\mathbf{e}2$ in particular) was not necessarily to be expected. In fact, SentiWordNet 1.0 was built by a semi-supervised learning method that uses vectors $\mathbf{e}2$ as its only initial training data. This paper thus shows that, starting from $\mathbf{e}2$ as the only manually annotated data, the best results are obtained neither by the semi-supervised method that generated SentiWordNet 1.0, nor by PageRank, but by the concatenation of the former with the latter.

		Positivity		Negativity	
$\mathbf{e}$	PageRank?	τ_p	Δ	τ_p	Δ
$\mathbf{e}1$	before	.500		.500	
	after	.496	(-0.81%)	.549	(9.83%)
$\mathbf{e}2$	before	.500		.500	
	after	.467	(-6.65%)	.502	(0.31%)
$\mathbf{e}3$	before	.500		.500	
	after	.471	(-5.79%)	.495	(-0.92%)
$\mathbf{e}4$	before	.349		.296	
	after	**.325**	(-6.75%)	**.284**	(-4.31%)
$\mathbf{e}5$	before	.400		.407	
	after	.380	(-4.88%)	.393	(-3.45%)

Table 1: Values of τ_p between predicted rankings and gold standard rankings (smaller is better). For each experiment the first line indicates the ranking obtained from the original $\mathbf{e}$ vector (before the application of PageRank), while the second line indicates the ranking obtained after the application of PageRank, with the relative improvement (a negative percentage indicates improvement).

6 Conclusions

We have investigated the applicability of a random-walk model to the problem of ranking synsets according to positivity and negativity. However, we conjecture that this model can be of more general use, i.e., for the determination of other properties of term senses, such as membership in a domain. This paper thus presents a proof-of-concept of the model, and the results of experiments support our intuitions.

Also, we see this work as a proof of concept for the applicability of general random-walk algorithms (and not just PageRank) to the determination of the semantic properties of synsets. In a more recent paper (Esuli and Sebastiani, 2007a) we have investigated a related random-walk model, one in which, symmetrically to the intuitions of the model presented in this paper, semantics flows from the *definiens* to the *definiendum*; a metaphor that proves no less powerful than the one we have championed in this paper.

References

Alina Andreevskaia and Sabine Bergler. 2006a. Mining WordNet for fuzzy sentiment: Sentiment tag extraction from WordNet glosses. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL'06)*, pages 209–216, Trento, IT.

Alina Andreevskaia and Sabine Bergler. 2006b. Sentiment tag extraction from WordNet glosses. In *Proceedings of*

the 5th Conference on Language Resources and Evaluation (LREC'06), Genova, IT.

Monica Bianchini, Marco Gori, and Franco Scarselli. 2005. Inside PageRank. *ACM Transactions on Internet Technology*, 5(1):92–128.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117.

Sabrina Cerini, Valentina Compagnoni, Alice Demontis, Maicol Formentelli, and Caterina Gandini. 2007. Micro-WNOp: A gold standard for the evaluation of automatically compiled lexical resources for opinion mining. In Andrea Sansò, editor, *Language resources and linguistic theory: Typology, second language acquisition, English linguistics*. Franco Angeli Editore, Milano, IT. Forthcoming.

Andrea Esuli and Fabrizio Sebastiani. 2005. Determining the semantic orientation of terms through gloss analysis. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM'05)*, pages 617–624, Bremen, DE.

Andrea Esuli and Fabrizio Sebastiani. 2006a. Determining term subjectivity and term orientation for opinion mining. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL'06)*, pages 193–200, Trento, IT.

Andrea Esuli and Fabrizio Sebastiani. 2006b. 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, Genova, IT.

Andrea Esuli and Fabrizio Sebastiani. 2007a. Random-walk models of term semantics: An application to opinion-related properties. Technical Report ISTI-009/2007, Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale dellle Ricerche, Pisa, IT.

Andrea Esuli and Fabrizio Sebastiani. 2007b. SENTIWORDNET: A high-coverage lexical resource for opinion mining. Technical Report 2007-TR-02, Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, IT.

Ronald Fagin, Ravi Kumar, Mohammad Mahdiany, D. Sivakumar, and Erik Veez. 2004. Comparing and aggregating rankings with ties. In *Proceedings of ACM International Conference on Principles of Database Systems (PODS'04)*, pages 47–58, Paris, FR.

Gregory Grefenstette, Yan Qu, David A. Evans, and James G. Shanahan. 2006. Validating the coverage of lexical resources for affect analysis and automatically classifying new words along semantic axes. In James G. Shanahan, Yan Qu, and Janyce Wiebe, editors, *Computing Attitude and Affect in Text: Theories and Applications*, pages 93–107. Springer, Heidelberg, DE.

Sanda H. Harabagiu, George A. Miller, and Dan I. Moldovan. 1999. WordNet 2: A morphologically and semantically enhanced resource. In *Proceedings of the ACL SIGLEX Workshop on Standardizing Lexical Resources*, pages 1–8, College Park, US.

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.

Taher H. Haveliwala. 2003. Topic-sensitive PageRank: A context-sensitive ranking algorithm for Web search. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):784–796.

Nancy Ide. 2006. Making senses: Bootstrapping sense-tagged lists of semantically-related words. In *Proceedings of the 7th International Conference on Computational Linguistics and Intelligent Text Processing (CICLING'06)*, pages 13–27, Mexico City, MX.

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)*, volume IV, pages 1115–1118, Lisbon, PT.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING'04)*, pages 1367–1373, Geneva, CH.

Rada Mihalcea. 2006. Random walks on text structures. In *Proceedings of the 7th International Conference on Computational Linguistics and Intelligent Text Processing (CICLING'06)*, pages 249–262, Mexico City, MX.

Pero Subasic and Alison Huettner. 2001. Affect analysis of text using fuzzy semantic typing. *IEEE Transactions on Fuzzy Systems*, 9(4):483–496.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting emotional polarity of words using spin model. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 133–140, Ann Arbor, US.

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.

Janyce Wiebe and Rada Mihalcea. 2006. Word sense and subjectivity. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL'06)*, pages 1065–1072, Sydney, AU.

Structured Models for Fine-to-Coarse Sentiment Analysis

Ryan McDonald* **Kerry Hannan** **Tyler Neylon** **Mike Wells** **Jeff Reynar**
Google, Inc.
76 Ninth Avenue
New York, NY 10011
*Contact email: ryanmcd@google.com

Abstract

In this paper we investigate a structured model for jointly classifying the sentiment of text at varying levels of granularity. Inference in the model is based on standard sequence classification techniques using constrained Viterbi to ensure consistent solutions. The primary advantage of such a model is that it allows classification decisions from one level in the text to influence decisions at another. Experiments show that this method can significantly reduce classification error relative to models trained in isolation.

1 Introduction

Extracting sentiment from text is a challenging problem with applications throughout Natural Language Processing and Information Retrieval. Previous work on sentiment analysis has covered a wide range of tasks, including polarity classification (Pang et al., 2002; Turney, 2002), opinion extraction (Pang and Lee, 2004), and opinion source assignment (Choi et al., 2005; Choi et al., 2006). Furthermore, these systems have tackled the problem at different levels of granularity, from the document level (Pang et al., 2002), sentence level (Pang and Lee, 2004; Mao and Lebanon, 2006), phrase level (Turney, 2002; Choi et al., 2005), as well as the speaker level in debates (Thomas et al., 2006). The ability to classify sentiment on multiple levels is important since different applications have different needs. For example, a summarization system for product reviews might require polarity classification at the sentence or phrase level; a question answering system would most likely require the sentiment of paragraphs; and a system that determines which articles from an online news source are editorial in nature would require a document level analysis.

This work focuses on models that jointly classify sentiment on multiple levels of granularity. Consider the following example,

> *This is the first Mp3 player that I have used ... I thought it sounded great ... After only a few weeks, it started having trouble with the earphone connection ... I won't be buying another.*
>
> Mp3 player review from Amazon.com

This excerpt expresses an overall negative opinion of the product being reviewed. However, not all parts of the review are negative. The first sentence merely provides some context on the reviewer's experience with such devices and the second sentence indicates that, at least in one regard, the product performed well. We call the problem of identifying the sentiment of the document and of all its subcomponents, whether at the paragraph, sentence, phrase or word level, *fine-to-coarse sentiment analysis*.

The simplest approach to fine-to-coarse sentiment analysis would be to create a separate system for each level of granularity. There are, however, obvious advantages to building a single model that classifies each level in tandem. Consider the sentence,

> *My 11 year old daughter has also been using it and it is a lot harder than it looks.*

In isolation, this sentence appears to convey negative sentiment. However, it is part of a favorable review

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 432–439,
Prague, Czech Republic, June 2007.

for a piece of fitness equipment, where *hard* essentially means *good workout*. In this domain, *hard*'s sentiment can only be determined in context (i.e., *hard* to assemble versus a *hard* workout). If the classifier knew the overall sentiment of a document, then disambiguating such cases would be easier.

Conversely, document level analysis can benefit from finer level classification by taking advantage of common discourse cues, such as the last sentence being a reliable indicator for overall sentiment in reviews. Furthermore, during training, the model will not need to modify its parameters to explain phenomena like the typically positive word *great* appearing in a negative text (as is the case above). The model can also avoid overfitting to features derived from neutral or objective sentences. In fact, it has already been established that sentence level classification can improve document level analysis (Pang and Lee, 2004). This line of reasoning suggests that a cascaded approach would also be insufficient. Valuable information is passed in both directions, which means any model of fine-to-coarse analysis should account for this.

In Section 2 we describe a simple structured model that jointly learns and infers sentiment on different levels of granularity. In particular, we reduce the problem of joint sentence and document level analysis to a sequential classification problem using constrained Viterbi inference. Extensions to the model that move beyond just two-levels of analysis are also presented. In Section 3 an empirical evaluation of the model is given that shows significant gains in accuracy over both single level classifiers and cascaded systems.

1.1 Related Work

The models in this work fall into the broad class of global structured models, which are typically trained with structured learning algorithms. Hidden Markov models (Rabiner, 1989) are one of the earliest structured learning algorithms, which have recently been followed by discriminative learning approaches such as conditional random fields (CRFs) (Lafferty et al., 2001; Sutton and McCallum, 2006), the structured perceptron (Collins, 2002) and its large-margin variants (Taskar et al., 2003; Tsochantaridis et al., 2004; McDonald et al., 2005; Daumé III et al., 2006). These algorithms are usually applied to sequential labeling or chunking, but have also been applied to parsing (Taskar et al., 2004; McDonald et al., 2005), machine translation (Liang et al., 2006) and summarization (Daumé III et al., 2006).

Structured models have previously been used for sentiment analysis. Choi et al. (2005, 2006) use CRFs to learn a global sequence model to classify and assign sources to opinions. Mao and Lebanon (2006) used a sequential CRF regression model to measure polarity on the sentence level in order to determine the *sentiment flow* of authors in reviews. Here we show that fine-to-coarse models of sentiment can often be reduced to the sequential case.

Cascaded models for fine-to-coarse sentiment analysis were studied by Pang and Lee (2004). In that work an initial model classified each sentence as being subjective or objective using a global min-cut inference algorithm that considered local labeling consistencies. The top subjective sentences are then input into a standard document level polarity classifier with improved results. The current work differs from that in Pang and Lee through the use of a single joint structured model for both sentence and document level analysis.

Many problems in natural language processing can be improved by learning and/or predicting multiple outputs jointly. This includes parsing and relation extraction (Miller et al., 2000), entity labeling and relation extraction (Roth and Yih, 2004), and part-of-speech tagging and chunking (Sutton et al., 2004). One interesting work on sentiment analysis is that of Popescu and Etzioni (2005) which attempts to classify the sentiment of phrases with respect to possible product features. To do this an iterative algorithm is used that attempts to globally maximize the classification of all phrases while satisfying local consistency constraints.

2 Structured Model

In this section we present a structured model for fine-to-coarse sentiment analysis. We start by examining the simple case with two-levels of granularity – the sentence and document – and show that the problem can be reduced to sequential classification with constrained inference. We then discuss the feature space and give an algorithm for learning the parameters based on large-margin structured learning.

Extensions to the model are also examined.

2.1 A Sentence-Document Model

Let $\mathcal{Y}(d)$ be a discrete set of sentiment labels at the document level and $\mathcal{Y}(s)$ be a discrete set of sentiment labels at the sentence level. As input a system is given a document containing sentences $\boldsymbol{s} = s_1, \ldots, s_n$ and must produce sentiment labels for the document, $y^d \in \mathcal{Y}(d)$, and each individual sentence, $\boldsymbol{y}^s = y^s_1, \ldots, y^s_n$, where $y^s_i \in \mathcal{Y}(s)\ \forall\ 1 \leq i \leq n$. Define $\boldsymbol{y} = (y^d, \boldsymbol{y}^s) = (y^d, y^s_1, \ldots, y^s_n)$ as the joint labeling of the document and sentences. For instance, in Pang and Lee (2004), y^d would be the polarity of the document and y^s_i would indicate whether sentence s_i is subjective or objective. The models presented here are compatible with arbitrary sets of discrete output labels.

Figure 1 presents a model for jointly classifying the sentiment of both the sentences and the document. In this undirected graphical model, the label of each sentence is dependent on the labels of its neighbouring sentences plus the label of the document. The label of the document is dependent on the label of every sentence. Note that the edges between the input (each sentence) and the output labels are not solid, indicating that they are given as input and are not being modeled. The fact that the sentiment of sentences is dependent not only on the local sentiment of other sentences, but also the global document sentiment – and vice versa – allows the model to directly capture the importance of classification decisions across levels in fine-to-coarse sentiment analysis. The local dependencies between sentiment labels on sentences is similar to the work of Pang and Lee (2004) where soft local consistency constraints were created between every sentence in a document and inference was solved using a min-cut algorithm. However, jointly modeling the document label and allowing for non-binary labels complicates min-cut style solutions as inference becomes intractable.

Learning and inference in undirected graphical models is a well studied problem in machine learning and NLP. For example, CRFs define the probability over the labels conditioned on the input using the property that the joint probability distribution over the labels factors over clique potentials in undirected graphical models (Lafferty et al., 2001).

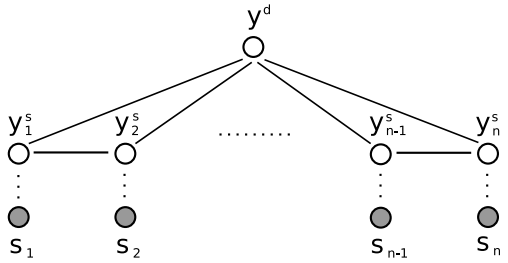


Figure 1: Sentence and document level model.

In this work we will use structured linear classifiers (Collins, 2002). We denote the score of a labeling $\boldsymbol{y}$ for an input $\boldsymbol{s}$ as $score(\boldsymbol{y}, \boldsymbol{s})$ and define this score as the sum of scores over each clique,

$$\begin{aligned} score(\boldsymbol{y}, \boldsymbol{s}) &= score((y^d, \boldsymbol{y}^s), \boldsymbol{s}) \\ &= score((y^d, y^s_1, \ldots, y^s_n), \boldsymbol{s}) \\ &= \sum_{i=2}^{n} score(y^d, y^s_{i-1}, y^s_i, \boldsymbol{s}) \end{aligned}$$

where each clique score is a linear combination of features and their weights,

$$score(y^d, y^s_{i-1}, y^s_i, \boldsymbol{s}) = \mathbf{w} \cdot \mathbf{f}(y^d, y^s_{i-1}, y^s_i, \boldsymbol{s}) \quad (1)$$

and $\mathbf{f}$ is a high dimensional feature representation of the clique and $\mathbf{w}$ a corresponding weight vector. Note that $\boldsymbol{s}$ is included in each score since it is given as input and can always be conditioned on.

In general, inference in undirected graphical models is intractable. However, for the common case of sequences (a.k.a. linear-chain models) the Viterbi algorithm can be used (Rabiner, 1989; Lafferty et al., 2001). Fortunately there is a simple technique that reduces inference in the above model to sequence classification with a constrained version of Viterbi.

2.1.1 Inference as Sequential Labeling

The inference problem is to find the highest scoring labeling $\boldsymbol{y}$ for an input $\boldsymbol{s}$, i.e.,

$$\arg\max_{\boldsymbol{y}} \ score(\boldsymbol{y}, \boldsymbol{s})$$

If the document label y^d is fixed, then inference in the model from Figure 1 reduces to the sequential case. This is because the search space is only over the sentence labels y^s_i, whose graphical structure forms a chain. Thus the problem of finding the

Input: $\boldsymbol{s} = s_1, \ldots, s_n$
1. $\boldsymbol{y}$ = null
2. for each $y^d \in \mathcal{Y}(d)$
3. $\quad \boldsymbol{y}^s = \arg\max_{\boldsymbol{y}^s} \; score((y^d, \boldsymbol{y}^s), \boldsymbol{s})$
4. $\quad \boldsymbol{y}' = (y^d, \boldsymbol{y}^s)$
5. $\quad$ if $score(\boldsymbol{y}', \boldsymbol{s}) > score(\boldsymbol{y}, \boldsymbol{s})$ or $\boldsymbol{y}$ = null
6. $\quad\quad \boldsymbol{y} = \boldsymbol{y}'$
7. return $\boldsymbol{y}$

Figure 2: Inference algorithm for model in Figure 1. The argmax in line 3 can be solved using Viterbi's algorithm since y^d is fixed.

highest scoring sentiment labels for all sentences, given a particular document label y^d, can be solved efficiently using Viterbi's algorithm.

The general inference problem can then be solved by iterating over each possible y^d, finding $\boldsymbol{y^s}$ maximizing $score((y^d, \boldsymbol{y^s}), \boldsymbol{s})$ and keeping the single best $\boldsymbol{y} = (y^d, \boldsymbol{y^s})$. This algorithm is outlined in Figure 2 and has a runtime of $O(|\mathcal{Y}(d)||\mathcal{Y}(s)|^2 n)$, due to running Viterbi $|\mathcal{Y}(d)|$ times over a label space of size $|\mathcal{Y}(s)|$. The algorithm can be extended to produce exact k-best lists. This is achieved by using k-best Viterbi techniques to return the k-best global labelings for each document label in line 3. Merging these sets will produce the final k-best list.

It is possible to view the inference algorithm in Figure 2 as a constrained Viterbi search since it is equivalent to flattening the model in Figure 1 to a sequential model with sentence labels from the set $\mathcal{Y}(s) \times \mathcal{Y}(d)$. The resulting Viterbi search would then need to be constrained to ensure consistent solutions, i.e., the label assignments agree on the document label over all sentences. If viewed this way, it is also possible to run a constrained forward-backward algorithm and learn the parameters for CRFs as well.

2.1.2 Feature Space

In this section we define the feature representation for each clique, $\mathbf{f}(y^d, y^s_{i-1}, y^s_i, \boldsymbol{s})$. Assume that each sentence s_i is represented by a set of binary predicates $\mathcal{P}(s_i)$. This set can contain any predicate over the input $\boldsymbol{s}$, but for the present purposes it will include all the unigram, bigram and trigrams in the sentence s_i conjoined with their part-of-speech (obtained from an automatic classifier). Back-offs of each predicate are also included where one or more word is discarded. For instance, if $\mathcal{P}(s_i)$ contains the predicate *a:DT_great:JJ_product:NN*, then it would also have the predicates *a:DT_great:JJ_*:NN*, *a:DT_*:JJ_product:NN*, **:DT_great:JJ_product:NN*, *a:DT_*:JJ_*:NN*, etc. Each predicate, p, is then conjoined with the label information to construct a binary feature. For example, if the sentence label set is $\mathcal{Y}(s) = \{\text{subj}, \text{obj}\}$ and the document set is $\mathcal{Y}(d) = \{\text{pos}, \text{neg}\}$, then the system might contain the following feature,

$$\mathbf{f}_{(j)}(y^d, y^s_{i-1}, y^s_i, \boldsymbol{s}) = \begin{cases} 1 & \text{if } p \in \mathcal{P}(s_i) \\ & \text{and } y^s_{i-1} = \text{obj} \\ & \text{and } y^s_i = \text{subj} \\ & \text{and } y^d = \text{neg} \\ 0 & \text{otherwise} \end{cases}$$

Where $\mathbf{f}_{(j)}$ is the j^{th} dimension of the feature space. For each feature, a set of back-off features are included that only consider the document label y^d, the current sentence label y^s_i, the current sentence and document label y^s_i and y^d, and the current and previous sentence labels y^s_i and y^s_{i-1}. Note that through these back-off features the joint models feature set will subsume the feature set of any individual level model. Only features observed in the training data were considered. Depending on the data set, the dimension of the feature vector $\mathbf{f}$ ranged from 350K to 500K. Though the feature vectors can be sparse, the feature weights will be learned using large-margin techniques that are well known to be robust to large and sparse feature representations.

2.1.3 Training the Model

Let $\mathcal{Y} = \mathcal{Y}(d) \times \mathcal{Y}(s)^n$ be the set of all valid sentence-document labelings for an input $\boldsymbol{s}$. The weights, $\mathbf{w}$, are set using the MIRA learning algorithm, which is an inference based online large-margin learning technique (Crammer and Singer, 2003; McDonald et al., 2005). An advantage of this algorithm is that it relies only on inference to learn the weight vector (see Section 2.1.1). MIRA has been shown to provide state-of-the-art accuracy for many language processing tasks including parsing, chunking and entity extraction (McDonald, 2006).

The basic algorithm is outlined in Figure 3. The algorithm works by considering a single training instance during each iteration. The weight vector $\mathbf{w}$ is updated in line 4 through a quadratic programming problem. This update modifies the weight vector so

Training data: $\mathcal{T} = \{(\boldsymbol{y}_t, \boldsymbol{s}_t)\}_{t=1}^{T}$

1. $\mathbf{w}^{(0)} = 0;\ i = 0$
2. for $n : 1..N$
3. for $t : 1..T$
4. $\mathbf{w}^{(i+1)} = \arg\min_{\mathbf{w}^*} \left\|\mathbf{w}^* - \mathbf{w}^{(i)}\right\|$
   s.t. $score(\boldsymbol{y}_t, \boldsymbol{s}_t) - score(\boldsymbol{y}', \boldsymbol{s}) \geq L(\boldsymbol{y}_t, \boldsymbol{y}')$
   relative to $\mathbf{w}^*$
   $\forall \boldsymbol{y}' \in \mathcal{C} \subset \mathcal{Y}$, where $|\mathcal{C}| = k$
5. $i = i + 1$
6. return $\mathbf{w}^{(N \times T)}$

Figure 3: MIRA learning algorithm.

that the score of the correct labeling is larger than the score of every labeling in a constraint set $\mathcal{C}$ with a margin proportional to the loss. The constraint set $\mathcal{C}$ can be chosen arbitrarily, but it is usually taken to be the k labelings that have the highest score under the old weight vector $\mathbf{w}^{(i)}$ (McDonald et al., 2005). In this manner, the learning algorithm can update its parameters relative to those labelings closest to the decision boundary. Of all the weight vectors that satisfy these constraints, MIRA chooses the one that is as close as possible to the previous weight vector in order to retain information about previous updates.

The loss function $L(\boldsymbol{y}, \boldsymbol{y}')$ is a positive real valued function and is equal to zero when $\boldsymbol{y} = \boldsymbol{y}'$. This function is task specific and is usually the hamming loss for sequence classification problems (Taskar et al., 2003). Experiments with different loss functions for the joint sentence-document model on a development data set indicated that the hamming loss over sentence labels multiplied by the 0-1 loss over document labels worked best.

An important modification that was made to the learning algorithm deals with how the k constraints are chosen for the optimization. Typically these constraints are the k highest scoring labelings under the current weight vector. However, early experiments showed that the model quickly learned to discard any labeling with an incorrect document label for the instances in the training set. As a result, the constraints were dominated by labelings that only differed over sentence labels. This did not allow the algorithm adequate opportunity to set parameters relative to incorrect document labeling decisions. To combat this, k was divided by the number of document labels, to get a new value k'. For each document label, the k' highest scoring labelings were

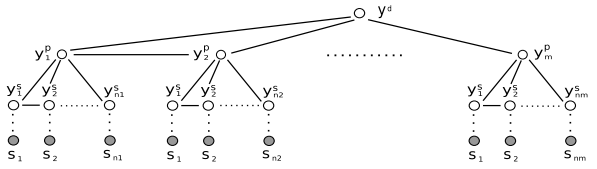


Figure 4: An extension to the model from Figure 1 incorporating paragraph level analysis.

extracted. Each of these sets were then combined to produce the final constraint set. This allowed constraints to be equally distributed amongst different document labels.

Based on performance on the development data set the number of training iterations was set to $N = 5$ and the number of constraints to $k = 10$. Weight averaging was also employed (Collins, 2002), which helped improve performance.

2.2 Beyond Two-Level Models

To this point, we have focused solely on a model for two-level fine-to-coarse sentiment analysis not only for simplicity, but because the experiments in Section 3 deal exclusively with this scenario. In this section, we briefly discuss possible extensions for more complex situations. For example, longer documents might benefit from an analysis on the paragraph level as well as the sentence and document levels. One possible model for this case is given in Figure 4, which essentially inserts an additional layer between the sentence and document level from the original model. Sentence level analysis is dependent on neighbouring sentences as well as the paragraph level analysis, and the paragraph analysis is dependent on each of the sentences within it, the neighbouring paragraphs, and the document level analysis. This can be extended to an arbitrary level of fine-to-coarse sentiment analysis by simply inserting new layers in this fashion to create more complex hierarchical models.

The advantage of using hierarchical models of this form is that they are nested, which keeps inference tractable. Observe that each pair of adjacent levels in the model is equivalent to the original model from Figure 1. As a result, the scores of the every label at each node in the graph can be calculated with a straight-forward bottom-up dynamic programming algorithm. Details are omitted

	Sentence Stats				Document Stats		
	Pos	Neg	Neu	Tot	Pos	Neg	Tot
Car	472	443	264	1179	98	80	178
Fit	568	635	371	1574	92	97	189
Mp3	485	464	214	1163	98	89	187
Tot	1525	1542	849	3916	288	266	554

Table 1: Data statistics for corpus. Pos = positive polarity, Neg = negative polarity, Neu = no polarity.

for space reasons.

Other models are possible where dependencies occur across non-neighbouring levels, e.g., by inserting edges between the sentence level nodes and the document level node. In the general case, inference is exponential in the size of each clique. Both the models in Figure 1 and Figure 4 have maximum clique sizes of three.

3 Experiments

3.1 Data

To test the model we compiled a corpus of 600 online product reviews from three domains: car seats for children, fitness equipment, and Mp3 players. Of the original 600 reviews that were gathered, we discarded duplicate reviews, reviews with insufficient text, and spam. All reviews were labeled by online customers as having a positive or negative polarity on the document level, i.e., $\mathcal{Y}(d) = \{\text{pos}, \text{neg}\}$. Each review was then split into sentences and every sentence annotated by a single annotator as either being positive, negative or neutral, i.e., $\mathcal{Y}(s) = \{\text{pos}, \text{neg}, \text{neu}\}$. Data statistics for the corpus are given in Table 1.

All sentences were annotated based on their context within the document. Sentences were annotated as neutral if they conveyed no sentiment or had indeterminate sentiment from their context. Many neutral sentences pertain to the circumstances under which the product was purchased. A common class of sentences were those containing product features. These sentences were annotated as having positive or negative polarity if the context supported it. This could include punctuation such as exclamation points, smiley/frowny faces, question marks, etc. The supporting evidence could also come from another sentence, e.g., *"I love it. It has 64Mb of memory and comes with a set of earphones"*.

3.2 Results

Three baseline systems were created,

- **Document-Classifier** is a classifier that learns to predict the document label only.
- **Sentence-Classifier** is a classifier that learns to predict sentence labels in isolation of one another, i.e., without consideration for either the document or neighbouring sentences sentiment.
- **Sentence-Structured** is another sentence classifier, but this classifier uses a sequential chain model to learn and classify sentences. The third baseline is essentially the model from Figure 1 without the top level document node. This baseline will help to gage the empirical gains of the different components of the joint structured model on sentence level classification.

The model described in Section 2 will be called **Joint-Structured**. All models use the same basic predicate space: unigram, bigram, trigram conjoined with part-of-speech, plus back-offs of these (see Section 2.1.2 for more). However, due to the structure of the model and its label space, the feature space of each might be different, e.g., the document classifier will only conjoin predicates with the document label to create the feature set. All models are trained using the MIRA learning algorithm.

Results for each model are given in the first four rows of Table 2. These results were gathered using 10-fold cross validation with one fold for development and the other nine folds for evaluation. This table shows that classifying sentences in isolation from one another is inferior to accounting for a more global context. A significant increase in performance can be obtained when labeling decisions between sentences are modeled (Sentence-Structured). More interestingly, even further gains can be had when document level decisions are modeled (Joint-Structured). In many cases, these improvements are highly statistically significant.

On the document level, performance can also be improved by incorporating sentence level decisions – though these improvements are not consistent. This inconsistency may be a result of the model overfitting on the small set of training data. We

suspect this because the document level error rate on the Mp3 training set converges to zero much more rapidly for the Joint-Structured model than the Document-Classifier. This suggests that the Joint-Structured model might be relying too much on the sentence level sentiment features – in order to minimize its error rate – instead of distributing the weights across all features more evenly.

One interesting application of sentence level sentiment analysis is summarizing product reviews on retail websites like Amazon.com or review aggregators like Yelp.com. In this setting the correct polarity of a document is often known, but we wish to label sentiment on the sentence or phrase level to aid in generating a cohesive and informative summary. The joint model can be used to classify sentences in this setting by constraining inference to the known fixed document label for a review. If this is done, then sentiment accuracy on the sentence level increases substantially from 62.6% to 70.3%.

Finally we should note that experiments using CRFs to train the structured models and logistic regression to train the local models yielded similar results to those in Table 2.

3.2.1 Cascaded Models

Another approach to fine-to-coarse sentiment analysis is to use a cascaded system. In such a system, a sentence level classifier might first be run on the data, and then the results input into a document level classifier – or vice-versa.[1] Two cascaded systems were built. The first uses the Sentence-Structured classifier to classify all the sentences from a review, then passes this information to the document classifier as input. In particular, for every predicate in the original document classifier, an additional predicate that specifies the polarity of the sentence in which this predicate occurred was created. The second cascaded system uses the document classifier to determine the global polarity, then passes this information as input into the Sentence-Structured model, constructing predicates in a similar manner.

The results for these two systems can be seen in the last two rows of Table 2. In both cases there is a slight improvement in performance suggesting that an iterative approach might be beneficial. That is, a system could start by classifying documents, use the document information to classify sentences, use the sentence information to classify documents, and repeat until convergence. However, experiments showed that this did not improve accuracy over a single iteration and often hurt performance.

Improvements from the cascaded models are far less consistent than those given from the joint structure model. This is because decisions in the cascaded system are passed to the next layer as the "gold" standard at test time, which results in errors from the first classifier propagating to errors in the second. This could be improved by passing a lattice of possibilities from the first classifier to the second with corresponding confidences. However, solutions such as these are really just approximations of the joint structured model that was presented here.

4 Future Work

One important extension to this work is to augment the models for partially labeled data. It is realistic to imagine a training set where many examples do not have every level of sentiment annotated. For example, there are thousands of online product reviews with labeled document sentiment, but a much smaller amount where sentences are also labeled. Work on learning with hidden variables can be used for both CRFs (Quattoni et al., 2004) and for inference based learning algorithms like those used in this work (Liang et al., 2006).

Another area of future work is to empirically investigate the use of these models on longer documents that require more levels of sentiment analysis than product reviews. In particular, the relative position of a phrase to a contrastive discourse connective or a cue phrase like "in conclusion" or "to summarize" may lead to improved performance since higher level classifications can learn to weigh information passed from these lower level components more heavily.

5 Discussion

In this paper we have investigated the use of a global structured model that learns to predict sentiment on different levels of granularity for a text. We de-

[1] Alternatively, decisions from the sentence classifier can guide which input is seen by the document level classifier (Pang and Lee, 2004).

	Sentence Accuracy				Document Accuracy			
	Car	Fit	Mp3	Total	Car	Fit	Mp3	Total
Document-Classifier	-	-	-	-	72.8	80.1	**87.2**	80.3
Sentence-Classifier	54.8	56.8	49.4	53.1	-	-	-	-
Sentence-Structured	60.5	61.4	55.7	58.8	-	-	-	-
Joint-Structured	**63.5***	**65.2****	**60.1****	**62.6****	**81.5***	**81.9**	85.0	**82.8**
Cascaded Sentence → Document	60.5	61.4	55.7	58.8	75.9	80.7	86.1	81.1
Cascaded Document → Sentence	59.7	61.0	58.3	59.5	72.8	80.1	87.2	80.3

Table 2: Fine-to-coarse sentiment accuracy. Significance calculated using McNemar's test between top two performing systems. *Statistically significant $p < 0.05$. **Statistically significant $p < 0.005$.

scribed a simple model for sentence-document analysis and showed that inference in it is tractable. Experiments show that this model obtains higher accuracy than classifiers trained in isolation as well as cascaded systems that pass information from one level to another at test time. Furthermore, extensions to the sentence-document model were discussed and it was argued that a nested hierarchical structure would be beneficial since it would allow for efficient inference algorithms.

References

Y. Choi, C. Cardie, E. Riloff, and S. Patwardhan. 2005. Identifying sources of opinions with conditional random fields and extraction patterns. In *Proc. HLT/EMNLP*.

Y. Choi, E. Breck, and C. Cardie. 2006. Joint extraction of entities and relations for opinion recognition. In *Proc. EMNLP*.

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

K. Crammer and Y. Singer. 2003. Ultraconservative online algorithms for multiclass problems. *JMLR*.

Hal Daumé III, John Langford, and Daniel Marcu. 2006. Search-based structured prediction. In Submission.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*.

P. Liang, A. Bouchard-Cote, D. Klein, and B. Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proc. ACL*.

Y. Mao and G. Lebanon. 2006. Isotonic conditional random fields and local sentiment flow. In *Proc. NIPS*.

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

R. McDonald. 2006. *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.

S. Miller, H. Fox, L.A. Ramshaw, and R.M. Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *Proc NAACL*, pages 226–233.

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

B. Pang, L. Lee, and S. Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *EMNLP*.

A. Popescu and O. Etzioni. 2005. Extracting product features and opinions from reviews. In *Proc. HLT/EMNLP*.

A. Quattoni, M. Collins, and T. Darrell. 2004. Conditional random fields for object recognition. In *Proc. NIPS*.

L. R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, February.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proc. CoNLL*.

C. Sutton and A. McCallum. 2006. An introduction to conditional random fields for relational learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press.

C. Sutton, K. Rohanimanesh, and A. McCallum. 2004. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *Proc. ICML*.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin Markov networks. In *Proc. NIPS*.

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

M. Thomas, B. Pang, and L. Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proc. EMNLP*.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. 2004. Support vector learning for interdependent and structured output spaces. In *Proc. ICML*.

P. Turney. 2002. Thumbs up or thumbs down? Sentiment orientation applied to unsupervised classification of reviews. In *EMNLP*.

Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification

John Blitzer **Mark Dredze** **Fernando Pereira**
Department of Computer and Information Science
University of Pennsylvania
{blitzer|mdredze|pereria@cis.upenn.edu}

Abstract

Automatic sentiment classification has been extensively studied and applied in recent years. However, sentiment is expressed differently in different domains, and annotating corpora for every possible domain of interest is impractical. We investigate domain adaptation for sentiment classifiers, focusing on online reviews for different types of products. First, we extend to sentiment classification the recently-proposed structural correspondence learning (SCL) algorithm, reducing the relative error due to adaptation between domains by an average of 30% over the original SCL algorithm and 46% over a supervised baseline. Second, we identify a measure of domain similarity that correlates well with the potential for adaptation of a classifier from one domain to another. This measure could for instance be used to select a small set of domains to annotate whose trained classifiers would transfer well to many other domains.

1 Introduction

Sentiment detection and classification has received considerable attention recently (Pang et al., 2002; Turney, 2002; Goldberg and Zhu, 2004). While movie reviews have been the most studied domain, sentiment analysis has extended to a number of new domains, ranging from stock message boards to congressional floor debates (Das and Chen, 2001; Thomas et al., 2006). Research results have been deployed industrially in systems that gauge market reaction and summarize opinion from Web pages, discussion boards, and blogs.

With such widely-varying domains, researchers and engineers who build sentiment classification systems need to collect and curate data for each new domain they encounter. Even in the case of market analysis, if automatic sentiment classification were to be used across a wide range of domains, the effort to annotate corpora for each domain may become prohibitive, especially since product features change over time. We envision a scenario in which developers annotate corpora for a small number of domains, train classifiers on those corpora, and then apply them to other similar corpora. However, this approach raises two important questions. First, it is well known that trained classifiers lose accuracy when the test data distribution is significantly different from the training data distribution [1]. Second, it is not clear which notion of domain similarity should be used to select domains to annotate that would be good proxies for many other domains.

We propose solutions to these two questions and evaluate them on a corpus of reviews for four different types of products from Amazon: books, DVDs, electronics, and kitchen appliances[2]. First, we show how to extend the recently proposed structural cor-

[1] For surveys of recent research on domain adaptation, see the ICML 2006 Workshop on Structural Knowledge Transfer for Machine Learning (http://gameairesearch.uta.edu/) and the NIPS 2006 Workshop on Learning when test and training inputs have different distribution (http://ida.first.fraunhofer.de/projects/different06/)

[2] The dataset will be made available by the authors at publication time.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 440–447,
Prague, Czech Republic, June 2007.

respondence learning (SCL) domain adaptation algorithm (Blitzer et al., 2006) for use in sentiment classification. A key step in SCL is the selection of *pivot features* that are used to link the source and target domains. We suggest selecting pivots based not only on their common frequency but also according to their mutual information with the source labels. For data as diverse as product reviews, SCL can sometimes misalign features, resulting in degradation when we adapt between domains. In our second extension we show how to correct misalignments using a very small number of labeled instances.

Second, we evaluate the $\mathcal{A}$-distance (Ben-David et al., 2006) between domains as measure of the loss due to adaptation from one to the other. The $\mathcal{A}$-distance can be measured from unlabeled data, and it was designed to take into account only divergences which affect classification accuracy. We show that it correlates well with adaptation loss, indicating that we can use the $\mathcal{A}$-distance to select a subset of domains to label as sources.

In the next section we briefly review SCL and introduce our new pivot selection method. Section 3 describes datasets and experimental method. Section 4 gives results for SCL and the mutual information method for selecting pivot features. Section 5 shows how to correct feature misalignments using a small amount of labeled target domain data. Section 6 motivates the $\mathcal{A}$-distance and shows that it correlates well with adaptability. We discuss related work in Section 7 and conclude in Section 8.

2 Structural Correspondence Learning

Before reviewing SCL, we give a brief illustrative example. Suppose that we are adapting from reviews of computers to reviews of cell phones. While many of the features of a good cell phone review are the same as a computer review – the words "excellent" and "awful" for example – many words are totally new, like "reception". At the same time, many features which were useful for computers, such as "dual-core" are no longer useful for cell phones.

Our key intuition is that even when "good-quality reception" and "fast dual-core" are completely distinct for each domain, if they both have high correlation with "excellent" and low correlation with "awful" on *unlabeled* data, then we can tentatively align them. After learning a classifier for computer reviews, when we see a cell-phone feature like "good-quality reception", we know it should behave in a roughly similar manner to "fast dual-core".

2.1 Algorithm Overview

Given labeled data from a source domain and unlabeled data from both source and target domains, SCL first chooses a set of m pivot features which occur frequently in both domains. Then, it models the correlations between the pivot features and all other features by training linear pivot predictors to predict occurrences of each pivot in the unlabeled data from both domains (Ando and Zhang, 2005; Blitzer et al., 2006). The ℓth pivot predictor is characterized by its weight vector $\mathbf{w}_\ell$; positive entries in that weight vector mean that a non-pivot feature (like "fast dual-core") is highly correlated with the corresponding pivot (like "excellent").

The pivot predictor column weight vectors can be arranged into a matrix $W = [\mathbf{w}_\ell]_{\ell=1}^{n}$. Let $\theta \in \mathbb{R}^{k \times d}$ be the top k left singular vectors of W (here d indicates the total number of features). These vectors are the principal predictors for our weight space. If we chose our pivot features well, then we expect these principal predictors to discriminate among positive and negative words in both domains.

At training and test time, suppose we observe a feature vector $\mathbf{x}$. We apply the projection $\theta\mathbf{x}$ to obtain k new real-valued features. Now we learn a predictor for the augmented instance $\langle \mathbf{x}, \theta\mathbf{x} \rangle$. If θ contains meaningful correspondences, then the predictor which uses θ will perform well in both source and target domains.

2.2 Selecting Pivots with Mutual Information

The efficacy of SCL depends on the choice of pivot features. For the part of speech tagging problem studied by Blitzer et al. (2006), frequently-occurring words in both domains were good choices, since they often correspond to function words such as prepositions and determiners, which are good indicators of parts of speech. This is not the case for sentiment classification, however. Therefore, we require that pivot features also be good predictors of the source label. Among those features, we then choose the ones with highest mutual information to the source label. Table 1 shows the set-symmetric

SCL, not SCL-MI	SCL-MI, not SCL
book one <num> so all very about they like good when	*a_must a_wonderful loved_it weak don't_waste awful highly_recommended and_easy*

Table 1: Top pivots selected by SCL, but not SCL-MI (left) and vice-versa (right)

differences between the two methods for pivot selection when adapting a classifier from books to kitchen appliances. We refer throughout the rest of this work to our method for selecting pivots as SCL-MI.

3 Dataset and Baseline

We constructed a new dataset for sentiment domain adaptation by selecting Amazon product reviews for four different product types: books, DVDs, electronics and kitchen appliances. Each review consists of a rating (0-5 stars), a reviewer name and location, a product name, a review title and date, and the review text. Reviews with rating > 3 were labeled positive, those with rating < 3 were labeled negative, and the rest discarded because their polarity was ambiguous. After this conversion, we had 1000 positive and 1000 negative examples for each domain, the same balanced composition as the polarity dataset (Pang et al., 2002). In addition to the labeled data, we included between 3685 (DVDs) and 5945 (kitchen) instances of unlabeled data. The size of the unlabeled data was limited primarily by the number of reviews we could crawl and download from the Amazon website. Since we were able to obtain labels for all of the reviews, we also ensured that they were balanced between positive and negative examples, as well.

While the polarity dataset is a popular choice in the literature, we were unable to use it for our task. Our method requires many unlabeled reviews and despite a large number of IMDB reviews available online, the extensive curation requirements made preparing a large amount of data difficult [3].

For classification, we use linear predictors on unigram and bigram features, trained to minimize the Huber loss with stochastic gradient descent (Zhang, 2004). On the polarity dataset, this model matches the results reported by Pang et al. (2002). When we report results with SCL and SCL-MI, we require that pivots occur in more than five documents in each domain. We set k, the number of singular vectors of the weight matrix, to 50.

[3]For a description of the construction of the polarity dataset, see `http://www.cs.cornell.edu/people/pabo/movie-review-data/`.

4 Experiments with SCL and SCL-MI

Each labeled dataset was split into a training set of 1600 instances and a test set of 400 instances. All the experiments use a classifier trained on the training set of one domain and tested on the test set of a possibly different domain. The baseline is a linear classifier trained without adaptation, while the gold standard is an in-domain classifier trained on the same domain as it is tested.

Figure 1 gives accuracies for all pairs of domain adaptation. The domains are ordered clockwise from the top left: books, DVDs, electronics, and kitchen. For each set of bars, the first letter is the source domain and the second letter is the target domain. The thick horizontal bars are the accuracies of the in-domain classifiers for these domains. Thus the first set of bars shows that the baseline achieves 72.8% accuracy adapting from DVDs to books. SCL-MI achieves 79.7% and the in-domain gold standard is 80.4%. We say that the *adaptation loss* for the baseline model is 7.6% and the adaptation loss for the SCL-MI model is 0.7%. The *relative reduction in error due to adaptation* of SCL-MI for this test is 90.8%.

We can observe from these results that there is a rough grouping of our domains. Books and DVDs are similar, as are kitchen appliances and electronics, but the two groups are different from one another. Adapting classifiers from books to DVDs, for instance, is easier than adapting them from books to kitchen appliances. We note that when transferring from kitchen to electronics, SCL-MI actually outperforms the in-domain classifier. This is possible since the unlabeled data may contain information that the in-domain classifier does not have access to.

At the beginning of Section 2 we gave examples of how features can change behavior across domains. The first type of behavior is when predictive features from the source domain are not predictive or do not appear in the target domain. The second is

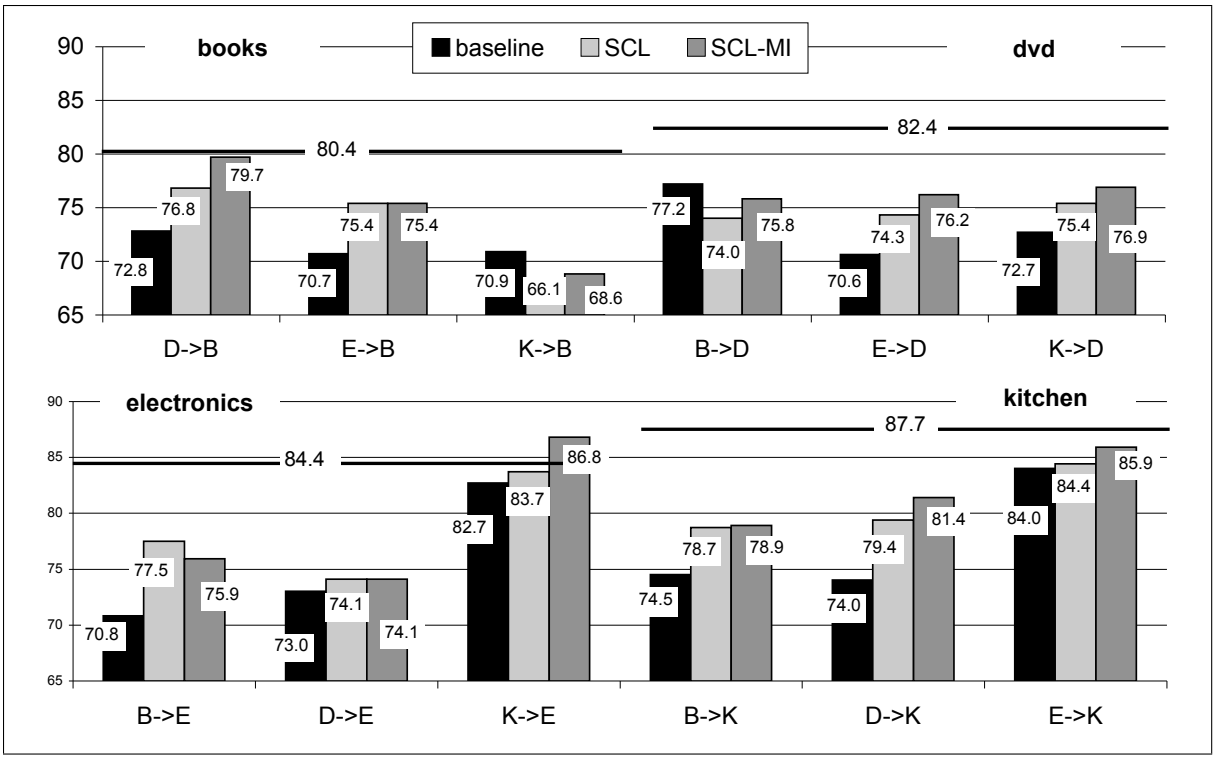


Figure 1: Accuracy results for domain adaptation between all pairs using SCL and SCL-MI. Thick black lines are the accuracies of in-domain classifiers.

domain\polarity	negative	positive
books	***plot <num>_pages predictable***	***reader grisham engaging***
	reading_this page_<num>	***must_read fascinating***
kitchen	***the_plastic poorly_designed***	***excellent_product espresso***
	leaking awkward_to defective	***are_perfect years_now a_breeze***

Table 2: Correspondences discovered by SCL for books and kitchen appliances. The top row shows features that only appear in books and the bottom features that only appear in kitchen appliances. The left and right columns show negative and positive features in correspondence, respectively.

when predictive features from the target domain do not appear in the source domain. To show how SCL deals with those domain mismatches, we look at the adaptation from book reviews to reviews of kitchen appliances. We selected the top 1000 most informative features in both domains. In both cases, between 85 and 90% of the informative features from one domain were not among the most informative of the other domain[4]. SCL addresses both of these issues simultaneously by aligning features from the two domains.

Table 2 illustrates one row of the projection matrix θ for adapting from books to kitchen appliances; the features on each row appear only in the corresponding domain. A supervised classifier trained on book reviews cannot assign weight to the kitchen features in the second row of table 2. In contrast, SCL assigns weight to these features indirectly through the projection matrix. When we observe the feature "predictable" with a negative book review, we update parameters corresponding to the entire projection, including the kitchen-specific features "poorly_designed" and "awkward_to".

While some rows of the projection matrix θ are

[4]There is a third type, features which are positive in one domain but negative in another, but they appear very infrequently in our datasets.

useful for classification, SCL can also misalign features. This causes problems when a projection is discriminative in the source domain but not in the target. This is the case for adapting from kitchen appliances to books. Since the book domain is quite broad, many projections in books model topic distinctions such as between religious and political books. These projections, which are uninformative as to the target label, are put into correspondence with the fewer discriminating projections in the much narrower kitchen domain. When we adapt from kitchen to books, we assign weight to these uninformative projections, degrading target classification accuracy.

5 Correcting Misalignments

We now show how to use a small amount of target domain labeled data to learn to ignore misaligned projections from SCL-MI. Using the notation of Ando and Zhang (2005), we can write the supervised training objective of SCL on the source domain as

$$\min_{\mathbf{w},\mathbf{v}} \sum_i L\left(\mathbf{w}'\mathbf{x}_i + \mathbf{v}'\theta\mathbf{x}_i, y_i\right) + \lambda||\mathbf{w}||^2 + \mu||\mathbf{v}||^2 \,,$$

where y is the label. The weight vector $\mathbf{w} \in \mathbb{R}^d$ weighs the original features, while $\mathbf{v} \in \mathbb{R}^k$ weighs the projected features. Ando and Zhang (2005) and Blitzer et al. (2006) suggest $\lambda = 10^{-4}, \mu = 0$, which we have used in our results so far.

Suppose now that we have trained source model weight vectors $\mathbf{w}_s$ and $\mathbf{v}_s$. A small amount of target domain data is probably insufficient to significantly change $\mathbf{w}$, but we can correct $\mathbf{v}$, which is much smaller. We augment each labeled target instance $\mathbf{x}_j$ with the label assigned by the source domain classifier (Florian et al., 2004; Blitzer et al., 2006). Then we solve

$$\min_{\mathbf{w},\mathbf{v}} \sum_j L\left(\mathbf{w}'\mathbf{x}_j + \mathbf{v}'\theta\mathbf{x}_j, y_j\right) + \lambda||\mathbf{w}||^2 + \mu||\mathbf{v} - \mathbf{v}_s||^2 \,.$$

Since we don't want to deviate significantly from the source parameters, we set $\lambda = \mu = 10^{-1}$.

Figure 2 shows the corrected SCL-MI model using 50 target domain labeled instances. We chose this number since we believe it to be a reasonable amount for a single engineer to label with minimal effort. For reasons of space, for each target domain we show adaptation from only the two domains on which SCL-MI performed the worst relative to the supervised baseline. For example, the book domain shows only results from electronics and kitchen, but not DVDs. As a baseline, we used the label of the source domain classifier as a feature in the target, but did not use any SCL features. We note that the baseline is very close to just using the source domain classifier, because with only 50 target domain instances we do not have enough data to relearn all of the parameters in $\mathbf{w}$. As we can see, though, relearning the 50 parameters in $\mathbf{v}$ is quite helpful. The corrected model *always* improves over the baseline for every possible transfer, including those not shown in the figure.

dom \ model	base	base +targ	scl	scl-mi	scl-mi +targ
books	8.9	9.0	7.4	5.8	**4.4**
dvd	8.9	8.9	7.8	6.1	**5.3**
electron	8.3	8.5	6.0	5.5	**4.8**
kitchen	10.2	9.9	7.0	5.6	**5.1**
average	9.1	9.1	7.1	5.8	**4.9**

Table 3: For each domain, we show the loss due to transfer for each method, averaged over all domains. The bottom row shows the average loss over all runs.

The idea of using the regularizer of a linear model to encourage the target parameters to be close to the source parameters has been used previously in domain adaptation. In particular, Chelba and Acero (2004) showed how this technique can be effective for capitalization adaptation. The major difference between our approach and theirs is that we only penalize deviation from the source parameters for the weights $\mathbf{v}$ of projected features, while they work with the weights of the original features only. For our small amount of labeled target data, attempting to penalize $\mathbf{w}$ using $\mathbf{w}_s$ performed no better than our baseline. Because we only need to learn to ignore projections that misalign features, we can make much better use of our labeled data by adapting only 50 parameters, rather than 200,000.

Table 3 summarizes the results of sections 4 and 5. Structural correspondence learning reduces the error due to transfer by 21%. Choosing pivots by mutual information allows us to further reduce the error to 36%. Finally, by adding 50 instances of target domain data and using this to correct the misaligned projections, we achieve an average relative

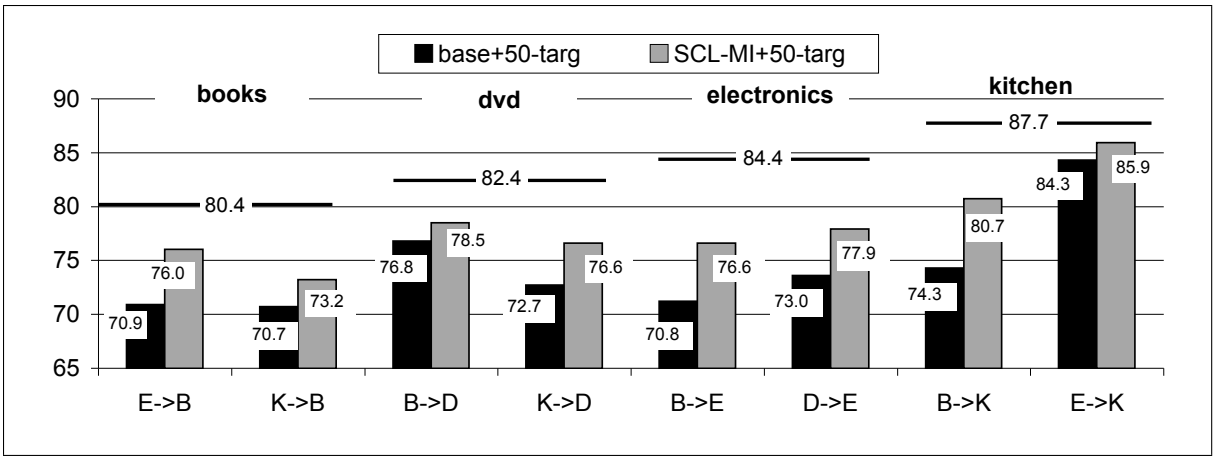


Figure 2: Accuracy results for domain adaptation with 50 labeled target domain instances.

reduction in error of 46%.

6 Measuring Adaptability

Sections 2-5 focused on how to adapt to a target domain when you had a labeled source dataset. We now take a step back to look at the problem of selecting source domain data to label. We study a setting where an engineer knows roughly her domains of interest but does not have any labeled data yet. In that case, she can ask the question "Which sources should I label to obtain the best performance over all my domains?" On our product domains, for example, if we are interested in classifying reviews of kitchen appliances, we know from sections 4-5 that it would be foolish to label reviews of books or DVDs rather than electronics. Here we show how to select source domains using only unlabeled data and the SCL representation.

6.1 The $\mathcal{A}$-distance

We propose to measure domain adaptability by using the divergence of two domains after the SCL projection. We can characterize domains by their induced distributions on instance space: the more different the domains, the more divergent the distributions. Here we make use of the $\mathcal{A}$-distance (Ben-David et al., 2006). The key intuition behind the $\mathcal{A}$-distance is that while two domains can differ in arbitrary ways, we are only interested in the differences that affect classification accuracy.

Let $\mathcal{A}$ be the family of subsets of $\mathbb{R}^k$ corresponding to characteristic functions of linear classifiers (sets on which a linear classifier returns positive value). Then the $\mathcal{A}$ distance between two probability distributions is

$$d_{\mathcal{A}}(\mathcal{D}, \mathcal{D}') = 2 \sup_{A \in \mathcal{A}} |\Pr_{\mathcal{D}}[A] - \Pr_{\mathcal{D}'}[A]| \ .$$

That is, we find the subset in $\mathcal{A}$ on which the distributions differ the most in the L_1 sense. Ben-David et al. (2006) show that computing the $\mathcal{A}$-distance for a finite sample is exactly the problem of minimizing the empirical risk of a classifier that discriminates between instances drawn from $\mathcal{D}$ and instances drawn from $\mathcal{D}'$. This is convenient for us, since it allows us to use classification machinery to compute the $\mathcal{A}$-distance.

6.2 Unlabeled Adaptability Measurements

We follow Ben-David et al. (2006) and use the Huber loss as a proxy for the $\mathcal{A}$-distance. Our procedure is as follows: Given two domains, we compute the SCL representation. Then we create a data set where each instance $\theta\mathbf{x}$ is labeled with the identity of the domain from which it came and train a linear classifier. For each pair of domains we compute the empirical average per-instance Huber loss, subtract it from 1, and multiply the result by 100. We refer to this quantity as the proxy $\mathcal{A}$-distance. When it is 100, the two domains are completely distinct. When it is 0, the two domains are indistinguishable using a linear classifier.

Figure 3 is a correlation plot between the proxy $\mathcal{A}$-distance and the adaptation error. Suppose we wanted to label two domains out of the four in such a

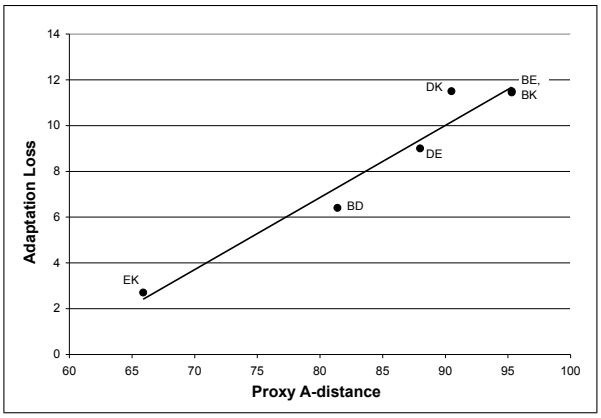


Figure 3: The proxy $\mathcal{A}$-distance between each domain pair plotted against the average adaptation loss of as measured by our baseline system. Each pair of domains is labeled by their first letters: EK indicates the pair electronics and kitchen.

way as to minimize our error on all the domains. Using the proxy $\mathcal{A}$-distance as a criterion, we observe that we would choose one domain from either books or DVDs, but not both, since then we would not be able to adequately cover electronics or kitchen appliances. Similarly we would also choose one domain from either electronics or kitchen appliances, but not both.

7 Related Work

Sentiment classification has advanced considerably since the work of Pang et al. (2002), which we use as our baseline. Thomas et al. (2006) use discourse structure present in congressional records to perform more accurate sentiment classification. Pang and Lee (2005) treat sentiment analysis as an ordinal ranking problem. In our work we only show improvement for the basic model, but all of these new techniques also make use of lexical features. Thus we believe that our adaptation methods could be also applied to those more refined models.

While work on domain adaptation for sentiment classifiers is sparse, it is worth noting that other researchers have investigated unsupervised and semisupervised methods for domain adaptation. The work most similar in spirit to ours that of Turney (2002). He used the difference in mutual information with two human-selected features (the words "excellent" and "poor") to score features in a completely unsupervised manner. Then he classified documents according to various functions of these mutual information scores. We stress that our method improves a supervised baseline. While we do not have a direct comparison, we note that Turney (2002) performs worse on movie reviews than on his other datasets, the same type of data as the polarity dataset.

We also note the work of Aue and Gamon (2005), who performed a number of empirical tests on domain adaptation of sentiment classifiers. Most of these tests were unsuccessful. We briefly note their results on combining a number of source domains. They observed that source domains closer to the target helped more. In preliminary experiments we confirmed these results. Adding more labeled data always helps, but diversifying training data does not. When classifying kitchen appliances, for any fixed amount of labeled data, it is always better to draw from electronics as a source than use some combination of all three other domains.

Domain adaptation alone is a generally well-studied area, and we cannot possibly hope to cover all of it here. As we noted in Section 5, we are able to significantly outperform basic structural correspondence learning (Blitzer et al., 2006). We also note that while Florian et al. (2004) and Blitzer et al. (2006) observe that including the label of a source classifier as a feature on small amounts of target data tends to improve over using either the source alone or the target alone, we did not observe that for our data. We believe the most important reason for this is that they explore structured prediction problems, where labels of surrounding words from the source classifier may be very informative, even if the current label is not. In contrast our simple binary prediction problem does not exhibit such behavior. This may also be the reason that the model of Chelba and Acero (2004) did not aid in adaptation.

Finally we note that while Blitzer et al. (2006) did combine SCL with labeled target domain data, they only compared using the label of SCL or non-SCL source classifiers as features, following the work of Florian et al. (2004). By only adapting the SCL-related part of the weight vector $\mathbf{v}$, we are able to make better use of our small amount of unlabeled data than these previous techniques.

8 Conclusion

Sentiment classification has seen a great deal of attention. Its application to many different domains of discourse makes it an ideal candidate for domain adaptation. This work addressed two important questions of domain adaptation. First, we showed that for a given source and target domain, we can significantly improve for sentiment classification the structural correspondence learning model of Blitzer et al. (2006). We chose pivot features using not only common frequency among domains but also mutual information with the source labels. We also showed how to correct structural correspondence misalignments by using a small amount of labeled target domain data.

Second, we provided a method for selecting those source domains most likely to adapt well to given target domains. The unsupervised $\mathcal{A}$-distance measure of divergence between domains correlates well with loss due to adaptation. Thus we can use the $\mathcal{A}$-distance to select source domains to label which will give low target domain error.

In the future, we wish to include some of the more recent advances in sentiment classification, as well as addressing the more realistic problem of ranking. We are also actively searching for a larger and more varied set of domains on which to test our techniques.

Acknowledgements

We thank Nikhil Dinesh for helpful advice throughout the course of this work. This material is based upon work partially supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD03001. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA or the Department of Interior-National BusinessCenter (DOI-NBC).

References

Rie Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*, 6:1817–1853.

Anthony Aue and Michael Gamon. 2005. Customizing sentiment classifiers to new domains: a case study. http://research.microsoft.com/ anthaue/.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2006. Analysis of representations for domain adaptation. In *Neural Information Processing Systems (NIPS)*.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Ciprian Chelba and Alex Acero. 2004. Adaptation of maximum entropy capitalizer: Little data can help a lot. In *EMNLP*.

Sanjiv Das and Mike Chen. 2001. Yahoo! for amazon: Extracting market sentiment from stock message boards. In *Proceedings of Athe Asia Pacific Finance Association Annual Conference*.

R. Florian, H. Hassan, A.Ittycheriah, H. Jing, N. Kambhatla, X. Luo, N. Nicolov, and S. Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *of HLT-NAACL*.

Andrew Goldberg and Xiaojin Zhu. 2004. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *HLT-NAACL 2006 Workshop on Textgraphs: Graph-based Algorithms for Natural Language Processing*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of Association for Computational Linguistics*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of Empirical Methods in Natural Language Processing*.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Peter Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of Association for Computational Linguistics*.

Tong Zhang. 2004. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *International Conference on Machine Learning (ICML)*.

Clustering Clauses for High-Level Relation Detection: An Information-theoretic Approach

Samuel Brody
School of Informatics
University of Edinburgh
s.brody@sms.ed.ac.uk

Abstract

Recently, there has been a rise of interest in unsupervised detection of high-level semantic relations involving complex units, such as phrases and whole sentences. Typically such approaches are faced with two main obstacles: data sparseness and correctly generalizing from the examples. In this work, we describe the *Clustered Clause* representation, which utilizes information-based clustering and inter-sentence dependencies to create a simplified and generalized representation of the grammatical clause. We implement an algorithm which uses this representation to detect a predefined set of high-level relations, and demonstrate our model's effectiveness in overcoming both the problems mentioned.

1 Introduction

The semantic relationship between words, and the extraction of meaning from syntactic data has been one of the main points of research in the field of computational linguistics (see Section 5 and references therein). Until recently, the focus has remained largely either at the single word or sentence level (for instance: dependency extraction, word-to-word semantic similarity from syntax, etc.) or on relations between units at a very high context level such as the entire paragraph or document (e.g. categorizing documents by topic).

Recently there have been several attempts to define frameworks for detecting and studying interactions at an intermediate context level, and involving whole clauses or sentences. Dagan et al. (2005) have emphasized the importance of detecting textual-entailment/implication between two sentences, and its place as a key component in many real-world applications, such as Information Retrieval and Question Answering.

When designing such a framework, one is faced with several obstacles. As we approach higher levels of complexity, the problem of defining the basic units we study (e.g. words, sentences etc.) and the increasing amount of interactions make the task very difficult. In addition, the data sparseness problem becomes more acute as the data units become more complex and have an increasing number of possible values, despite the fact that many of these values have similar or identical meaning.

In this paper we demonstrate an approach to solving the complexity and data sparseness problems in the task of detecting relations between sentences or clauses. We present the *Clustered Clause* structure, which utilizes information-based clustering and dependencies within the sentence to create a simplified and generalized representation of the grammatical clause and is designed to overcome both these problems.

The clustering method we employ is an integral part of the model. We evaluate our clusters against semantic similarity measures defined on the human-annotated WordNet structure (Fellbaum, 1998). The results of these comparisons show that our cluster members are very similar semantically. We also define a high-level relation detection task involving relations between clauses, evaluate our results, and demonstrate

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 448–455,
Prague, Czech Republic, June 2007.

the effectiveness of using our model in this task.

This work extends selected parts of Brody (2005), where further details can be found.

2 Model Construction

When designing our framework, we must address the complexity and sparseness problems encountered when dealing with whole sentences. Our solution to these issues combines two elements. First, to reduce complexity, we simplify a grammatical clause to its primary components - the subject, verb and object. Secondly, to provide a generalization framework which will enable us to overcome data-sparseness, we cluster each part of the clause using data from within the clause itself. By combining the simplified clause structure and the clustering we produce our *Clustered Clause* model - a triplet of clusters representing a generalized clause.

The Simplified Clause: In order to extract clauses from the text, we use Lin's parser MINIPAR (Lin, 1994). The output of the parser is a dependency tree of each sentence, also containing lemmatized versions of the component words. We extract the verb, subject and object of every clause (including subordinate clauses), and use this triplet of values, the simplified clause, in place of the original complete clause. As seen in Figure 1, these components make up the top (root) triangle of the clause parse tree. We also use the lemmatized form of the words provided by the parser, to further reduce complexity.

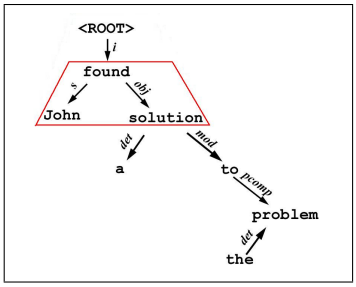


Figure 1: The parse tree for the sentence "John found a solution to the problem". The subject-verb-object triplet is marked with a border.

Clustering Clause Components: For our model, we cluster the data to provide both generalization, by using a cluster to represent a more generalized concept shared by its component words, and a form of dimensionality reduction, by using fewer units (clusters) to represent a much larger amount of words.

We chose to use the Sequential Information Bottleneck algorithm (Slonim et al., 2002) for our clustering tasks. The information Bottleneck principle views the clustering task as an optimization problem, where the clustering algorithm attempts to group together values of one variable while retaining as much information as possible regarding the values of another (target) variable. There is a trade-off between the compactness of the clustering and the amount of retained information. This algorithm (and others based on the IB principle) is especially suited for use with graphical models or dependency structures, since the distance measure it employs in the clustering is defined solely by the dependency relation between two variables, and therefore required no external parameters. The values of one variable are clustered using their co-occurrence distribution with regard to the values of the second (target) variable in the dependency relation. As an example, consider the following subject-verb co-occurrence matrix:

S \ V	**tell**	**scratch**	**drink**
dog	0	4	5
John	4	0	9
cat	0	6	3
man	6	1	2

The value in cell (i, j) indicates the number of times the noun i occurred as the subject of the verb j. Calculating the Mutual Information between the subjects variable (S) and verbs variable (V) in this table, we get $MI(S, V) = 0.52$ bits. Suppose we wish to cluster the subject nouns into two clusters while preserving the highest Mutual Information with regard to the verbs. The following co-occurrence matrix is the optimal clustering, and retains a M.I. value of 0.4 bits (77% of original):

Clustered S \ V	**tell**	**scratch**	**drink**
{dog,cat}	0	10	8
{John,man}	10	1	11

Notice that although the values in the *drink* column are higher than in others, and we may be

tempted to cluster together *dog* and *John* based on this column, the informativeness of this verb is smaller - if we know the verb is *tell* we can be sure the noun is not *dog* or *cat*, whereas if we know it is *drink*, we can only say it is slightly more probable that the noun is *John* or *dog*.

Our dependency structure consists of three variables: subject, verb, and object, and we take advantage of the subject-verb and verb-object dependencies in our clustering. The clustering was performed on each variable separately, in a two phase procedure (see Figure 2). In the first stage, we clustered the subject variable into 200 clusters[1], using the subject-verb dependency (i.e. the verb variable was the target). The same was done with the object variable, using the verb-object dependency. In the second phase, we wish to cluster the verb values with regard to both the subject and object variables. We could not use all pairs of subjects and objects values as the target variable in this task, since too many such combinations exist. Instead, we used a variable composed of all the pairs of subject and object *clusters* as the target for the verb clustering. In this fashion we produced 100 verb clusters.

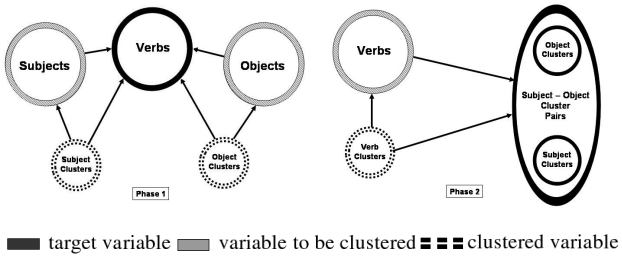


Figure 2: The two clustering phases. Arrows represent dependencies between the variables which are used in the clustering.

Combining the Model Elements: Having obtained our three clustered variables, our original simplified clause triplet can now be used to produce the *Clustered Clause* model. This model represents a clause in the data by a triplet of cluster indexes, one cluster index for each clustered variable. In order to map a clause in the text to its corresponding clustered clause, it is first parsed and lemmatized to obtain the subject, verb and object values, as described above, and then assigned to the clustered clause in which the subject cluster index is that of the cluster containing the subject word of the clause, and the same for the verb and object words. For example, the sentence "The terrorist threw the grenade" would be converted to the triplet (terrorist, throw, grenade) and assigned to the clustered clause composed of the three clusters to which these words belong. Other triplets assigned to this clustered clause might include (fundamentalist, throw, bomb) or (militant, toss, explosive). Applying this procedure to the entire text corpus results in a distillation of the text into a series of clustered clauses containing the essential information about the actions described in the text.

Technical Specifications: For this work we chose to use the entire Reuters Corpus (English, release 2000), containing 800,000 news articles collected uniformly from 20/8/1996 to 19/8/1997. Before clustering, several preprocessing steps were taken. We had a very large amount of word values for each of the Subject (85,563), Verb (4,593) and Object (74,842) grammatical categories. Many of the words were infrequent proper nouns or rare verbs and were of little interest in the pattern recognition task. We therefore removed the less frequent words - those appearing in their category less than one hundred times. We also cleaned our data by removing all words that were one letter in length, other than the word 'I'. These were mostly initials in names of people or companies, which were uninformative without the surrounding context. This processing step brought us to the final count of 2,874,763 clause triplets (75.8% of the original number), containing 3,153 distinct subjects, 1,716 distinct verbs, and 3,312 distinct objects. These values were clustered as described above. The clusters were used to convert the simplified clauses into clustered clauses.

3 Evaluating Cluster Quality

Examples of some of the resulting clusters are provided in Table 1. When manually examin-

[1]The chosen numbers of clusters are such that each the resulting clustered variables preserved approximately half of the co-occurrence mutual information that existed between the original (unclustered) variable and its target.

"Technical Developements" (Subject Cluster 160): treatment, drug, method, tactic, version, technology, software, design, device, vaccine, ending, tool, mechanism, technique, instrument, therapy, concept, model
"Ideals/Virtues" (Object Cluster 14): sovereignty, dominance, logic, validity, legitimacy, freedom, discipline, viability, referendum, wisdom, innocence, credential, integrity, independence
"Emphasis Verbs" (Verb Cluster 92): imply, signify, highlight, mirror, exacerbate, mark, signal, underscore, compound, precipitate, mask, illustrate, herald, reinforce, suggest, underline, aggravate, reflect, demonstrate, spell, indicate, deepen
"Plans" (Object Cluster 33): journey, arrangement, trip, effort, attempt, revolution, pullout, handover, sweep, preparation, filing, start, play, repatriation, redeployment, landing, visit, push, transition, process

Table 1: Example clusters (labeled manually).

ing the clusters, we noticed the "fine-tuning" of some of the clusters. For instance, we had a cluster of countries involved in military conflicts, and another for other countries; a cluster for winning game scores, and another for ties; etc. The fact that the algorithm separated these clusters indicates that the distinction between them is important with regard to the interactions within the clause. For instance, in the first example, the context in which countries from the first cluster appear is very different from that involving countries in the second cluster.

The effect of the dependencies we use is also strongly felt. Many clusters can be described by labels such as "things that are thrown" (*rock, flower, bottle, grenade* and others), or "verbs describing attacks" (*spearhead, foil, intensify, mount, repulse* and others). While such criteria may not be the first choice of someone who is asked to cluster verbs or nouns, they represent unifying themes which are very appropriate to pattern detection tasks, in which we wish to detect connections between actions described in the clauses. For instance, we would like to detect the relation between throwing and military/police action (much of the throwing described in the news reports fits this relation). In order to do this, we must have clusters which unite the words relevant to those actions. Other criteria for clustering would most likely not be suitable, since they would probably not put *egg, bottle* and *rock* in the same category. In this respect, our clustering method provides a more effective modeling of the domain knowledge.

3.1 Evaluation via Semantic Resource

Since the success of our pattern detection task depends to a large extent on the quality of our clusters, we performed an experiment designed to evaluate semantic similarity between members of our clusters. For this purpose we made use of the WordNet Similarity package (Pedersen et al., 2004). This package contains many similarity measures, and we selected three of them (Resnik (1995), Leacock and Chodorow (1997), Hirst and St-Onge (1997)), which make use of different aspects of WordNet (hierarchy and graph structure). We measured the average pairwise similarity between any two words appearing in the same cluster. We then performed the same calculation on a random grouping of the words, and compared the two scores. The results (Fig. 3) show that our clustering, based on co-occurrence statistics and dependencies within the sentence, correlates with a purely semantic similarity as represented by the WordNet structure, and cannot be attributed to chance.

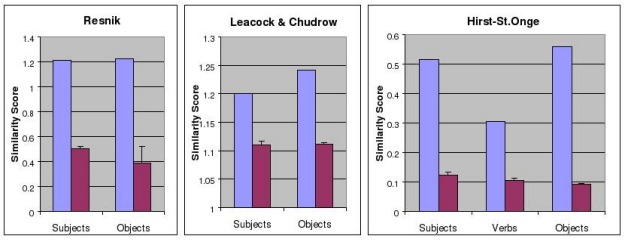


Figure 3: Inter-cluster similarity (average pairwise similarity between cluster members) in our clustering (light) and a random one (dark). Random clustering was performed 10 times. Average values are shown with error bars to indicate standard deviation. Only Hirst & St-Onge measure verb similarity.

4 Relation Detection Task

Motivation: In order to demonstrate the use of our model, we chose a relation detection task. The workshop on entailment mentioned in the introduction was mainly focused on detecting whether or not an entailment relation exists between two texts. In this work we present a com-

plementary approach - a method designed to automatically detect relations between portions of text and generate a knowledge base of the detected relations in a generalized form. As stated by (Dagan et al., 2005), such relations are important for IR applications. In addition, the patterns we employ are likely to be useful in other linguistic tasks involving whole clauses, such as paraphrase acquisition.

Pattern Definition: For our relation detection task, we searched for instances of predefined patterns indicating a relation between two clustered clauses. We restricted the search to clause pairs which co-occur within a distance of ten clauses[2] from each other. In addition to the distance restriction, we required an *anchor*: a noun that appears in both clauses, to further strengthen the relation between them. Noun anchors establish the fact that the two component actions described by the pattern involve the same entities, implying a direct connection between them. The use of verb anchors was also tested, but found to be less helpful in detecting significant patterns, since in most cases it simply found verbs which tend to repeat themselves frequently in a context. The method we describe assumes that statistically significant co-occurrences indicate a relationship between the clauses, but does not attempt to determine the type of relation.

Significance Calculation: The patterns detected by the system were scored using the statistical *p-value* measure. This value represents the probability of detecting a certain number of occurrences of a given pattern in the data under the independence assumption, i.e. assuming there is no connection between the two halves of the pattern. If the system has detected k instances of a certain pattern, we calculate the probability of encountering this number of instances under the independence assumption. The smaller the probability, the higher the significance. We consider patterns with a chance probability lower than 5% to be significant.

We assume a Gaussian-like distribution of occurrence probability for each pattern[3]. In order to estimate the mean and standard deviation values, we created 100 simulated sequences of triplets (representing clustered clauses) which were independently distributed and varied only in their overall probability of occurrence. We then estimated the mean and standard deviation for any pair of clauses in the actual data using the simulated sequences.

(X, VC_{36}, OC_7)	$\rightarrow_{10}$ (X, VC_{57}, OC_{85})
storm, lash, province	... storm, cross, Cuba
quake, shake, city	... quake, hit, Iran
earthquake, jolt, city	... earthquake, hit, Iran
(X, VC_{40}, OC_{165})	$\rightarrow_{10}$ (X, VC_{52}, OC_{152})
police, arrest, leader	... police, search, mosque
police, detain, leader	... police, search, mosque
police, arrest, member	... police, raid, enclave
(SC_{39}, VC_{21}, X)	$\rightarrow_{10}$ $(X, \textbf{\textit{beat}}\ ^4, OC_{155})$
sun, report, earnings	... earnings,beat,expectation
xerox, report, earnings	... earnings, beat, forecast
microsoft,release,result	... result, beat, forecast
(X, VC_{57}, OC_7)	$\rightarrow_{10}$ $(X, \textbf{\textit{cause}}\ ^4, OC_{153})$
storm, hit, coast	... storm, cause, damage
cyclone, near, coast	... cyclone, cause, damage
earthquake,hit,northwest	... earthquake,cause,damage
quake , hit, northwest	... quake, cause, casualty
earthquake,hit,city	... earthquake,cause,damage

Table 2: Example Patterns

4.1 Pattern Detection Results

In Table 2 we present several examples of high ranking (i.e. significance) patterns with different anchorings detected by our method. The detected patterns are represented using the notation of the form $(SC_i, VC_j, X) \rightarrow_n (X, VC_{i'}, OC_{j'})$. X indicates the anchoring word. In the example notation, the anchoring word is the object of the first clause and the subject of the second (O-S for short). n indicates the maximal distance between the two clauses. The terms SC, VC or OC with a subscripted index represent the cluster containing the subject, verb or object (respectively) of the appropriate clause. For instance, in the first example in Table 2, VC_{36} indicates verb cluster no. 36, containing the verbs *lash, shake* and *jolt*, among others.

[2]Our experiments showed that increasing the distance beyond this point did not result in significant increase in the number of detected patterns.

[3]Based on Gwadera et al. (2003), dealing with a similar, though simpler, case.

[4]In two of the patterns, instead of a cluster for the verb, we have a single word - *beat* or *cause*. This is the result of an automatic post-processing stage intended to prevent over-generalization. If all the instances of the pat-

Anchoring System	Number of Patterns Found
Subject-Subject	428
Object-Object	291
Subject-Object	180
Object-Subject	178

Table 3: Numbers of patterns found ($p < 5\%$)

Table 3 lists the number of patterns found, for each anchoring system. The different anchoring systems produce quantitatively different results. Anchoring between the same categories produces more patterns than between the same noun in different grammatical roles. This is expected, since many nouns can only play a certain part in the clause (for instance, many verbs cannot have an inanimate entity as their subject).

The number of instances of patterns we found for the anchored template might be considered low, and it is likely that some patterns were missed simply because their occurrence probability was very low and not enough instances of the pattern occurred in the text. In Section 4 we stated that in this task, we were more interested in precision than in recall. In order to detect a wider range of patterns, a less restricted definition of the patterns, or a different significance indicator, should be used (see Sec. 6).

Human Evaluation: In order to better determine the quality of patterns detected by our system, and confirm that the statistical significance testing is consistent with human judgment, we performed an evaluation experiment with the help of 22 human judges. We presented each of the judges with 60 example groups, 15 for each type of anchoring. Each example group contained three clause pairs conforming to the anchoring relation. The clauses were presented in a normalized form consisting only of a subject, object and verb converted to past tense, with the addition of necessary determiners and prepositions. For example, the triplet *(police, detain, leader)* was converted to *"The police detained the leader"*. In half the cases (randomly selected), these clause pairs were actual examples (instances) of a pattern detected by our system (instances group), such as those appearing in Table 2. In the other half, we listed three clause pairs, each of which conformed to the anchoring specification listed in Section 4, but which were randomly sampled from the data, and so had no connection to one another (baseline group). We asked the judges to rate on a scale of 1-5 whether they thought the clause pairs were a good set of examples of a common relation linking the first clause in each pair to the second one.

	Instances Score	Instances StdDev	Baseline Score	Baseline StdDev
All	3.5461	0.4780	2.6341	0.4244
O-S	3.9266	0.6058	2.8761	0.5096
O-O	3.4938	0.5144	2.7464	0.6205
S-O	3.4746	0.7340	2.5758	0.6314
S-S	3.2398	0.4892	2.3584	0.5645

Table 4: Results for human evaluation

Table 4 reports the overall average scores for baseline and instances groups, and for each of the four anchoring types individually. The scores were averaged over all examples and all judges. An ANOVA showed the difference in scores between the baseline and instance groups to be significant ($p < 0.001$) in all four cases.

Achievement of Model Goals: We employed clustering in our model to overcome data-sparseness. The importance of this decision was evident in our results. For example, the second pattern shown in Table 2 appeared only four times in the text. In these instances, verb cluster 40 was represented twice by the verb *arrest* and twice by *detain*. Two appearances are within the statistical deviation of all but the rarest words, and would not have been detected as significant without the clustering effect. This means the pattern would have been overlooked, despite the strongly intuitive connection it represents. The system detected several such patterns.

The other reason for clustering was generalization. Even in cases where patterns involving single words could have been detected, it would have been impossible to unify similar patterns into generalized ones. In addition, when encountering a new clause which differs slightly from

tern in the text contained the same word in a certain position (in these examples - the verb position in the second clause), this word was placed in that position in the generalized pattern, rather than the cluster it belonged to. Since we have no evidence for the fact that other words in the cluster can fit that position, using the cluster indicator would be over-generalizing.

the ones we recognized in the original data, there would be no way to recognize it and draw the appropriate conclusions. For example, there would be no way to relate the sentence *"The typhoon approached the coast"* to the fourth example pattern, and the connection with the resulting damage would not be recognized.

5 Comparison with Previous Work

The relationship between textual features and semantics and the use of syntax as an indicator of semantics has been widespread. Following the idea proposed in Harris' Distributional Hypothesis (Harris, 1985), that words occurring in similar contexts are semantically similar, many works have used different definitions of context to identify various types of semantic similarity. Hindle (1990) uses a mutual-information based metric derived from the distribution of subject, verb and object in a large corpus to classify nouns. Pereira et al. (1993) cluster nouns according to their distribution as direct objects of verbs, using information-theoretic tools (the predecessors of the tools we use in this work). They suggest that information theoretic measures can also measure semantic relatedness.

These works focus only on relatedness of individual words and do not describe how the automatic estimation of semantic similarity can be useful in real-world tasks. In our work we demonstrate that using clusters as generalized word units helps overcome the sparseness and generalization problems typically encountered when attempting to extract high-level patterns from text, as required for many applications.

The DIRT system (Lin and Pantel, 2001) deals with inference rules, and employs the notion of paths between two nouns in a sentence's parse tree. The system extracts such path structures from text, and provides a similarity measure between two such paths by comparing the words which fill the same slots in the two paths. After extracting the paths, the system finds groups of similar paths. This approach bears several similarities to the ideas described in this paper, since our structure can be seen as a specific path in the parse tree (probably the most basic one, see Fig. 1). In our setup, similar clauses are clustered together in the same *Clustered-Clause*, which could be compared to clustering DIRT's paths using its similarity measure. Despite these similarities, there are several important differences between the two systems. Our method uses only the relationships inside the path or clause in the clustering procedure, so the similarity is based on the structure itself. Furthermore, Lin and Pantel did not create path clusters or generalized paths, so that while their method allowed them to compare phrases for similarity, there is no convenient way to identify high level contextual relationships between two nearby sentences. This is one of the significant advantages that clustering has over similarity measures - it allows a group of similar objects to be represented by a single unit.

There have been several attempts to formulate and detect relationships at a higher context level. The VerbOcean project (Chklovski and Pantel, 2004) deals with relations between verbs. It presents an automatically acquired network of such relations, similar to the WordNet framework. Though the patterns used to acquire the relations are usually parts of a single sentence, the relationships themselves can also be used to describe connections between different sentences, especially the enablement and *happens-before* relations. Since verbs are the central part of the clause, VerbOcean can be viewed as detecting relations between clauses as whole units, as well as those between individual words. As a solution to the data sparseness problem, web queries are used. Torisawa (2006) addresses a similar problem, but focuses on temporal relations, and makes use of the phenomena of Japanese coordinate sentences. Neither of these approaches attempt to create generalized relations or group verbs into clusters, though in the case of VerbOcean this could presumably be done using the *similarity* and *strength* values which are defined and detected by the system.

6 Future Work

The clustered clause model presents many directions for further research. It may be productive to extend the model further, and include other parts of the sentence, such as adjectives

and adverbs. Clustering nouns by the adjectives that describe them may provide a more intuitive grouping. The addition of further elements to the structure may also allow the detection of new kinds of relations.

The news-oriented domain of the corpus we used strongly influenced our results. If we were interested in more mundane relations, involving day-to-day actions of individuals, a literary corpus would probably be more suitable.

In defining our pattern template, several elements were tailored specifically to our task. We limited ourselves to a very restricted set of patterns in order to better demonstrate the effectiveness of our model. For a more general knowledge acquisition task, several of these restrictions may be relaxed, allowing a much larger set of relations to be detected. For example, a less strict significance filter, such as the *support* and *confidence* measures commonly used in data mining, may be preferable. These can be set to different thresholds, according to the user's preference.

In our current work, in order to prevent overgeneralization, we employed a single step post-processing algorithm which detected the incorrect use of an entire cluster in place of a single word (see footnote for Table 2). This method allows only two levels of generalization - single words and whole clusters. A more appropriate way to handle generalization would be to use a hierarchical clustering algorithm. The Agglomerative Information Bottleneck (Slonim and Tishby, 1999) is an example of such an algorithm, and could be employed for this task. Use of a hierarchical method would result in several levels of clusters, representing different levels of generalization. It would be relatively easy to modify our procedure to reduce generalization to the level indicated by the pattern examples in the text, producing a more accurate description of the patterns detected.

Acknowledgments

The author acknowledges the support of EPSRC grant EP/C538447/1. The author would like to thank Naftali Tishby and Mirella Lapata for their supervision and assistance on large portions of the work presented here. I would also like to thank the anonymous reviewers and my friends and colleagues for their helpful comments.

References

Brody, Samuel. 2005. *Cluster-Based Pattern Recognition in Natural Language Text*. Master's thesis, Hebrew University, Jerusalem, Israel.

Chklovski, T. and P. Pantel. 2004. Verbocean: Mining the web for fine-grained semantic verb relations. In *Proc. of EMNLP*. pages 33–40.

Dagan, I., O. Glickman, and B. Magnini. 2005. The pascal recognising textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

Fellbaum, Christiane, editor. 1998. *WordNet: An Electronic Database*. MIT Press, Cambridge, MA.

Gwadera, R., M. Atallah, and W. Szpankowski. 2003. Reliable detection of episodes in event sequences. In *ICDM*.

Harris, Z. 1985. Distributional structure. *Katz, J. J. (ed.) The Philosophy of Linguistics* pages 26–47.

Hindle, Donald. 1990. Noun classification from predicate-argument structures. In *Meeting of the ACL*. pages 268–275.

Hirst, G. and D. St-Onge. 1997. Lexical chains as representation of context for the detection and correction of malapropisms. In *WordNet: An Electronic Lexical Database, ed., Christiane Fellbaum*. MIT Press.

Leacock, C. and M. Chodorow. 1997. Combining local context and wordnet similarity for word sense identification. In *WordNet: An Electronic Lexical Database, ed., Christiane Fellbaum*. MIT Press.

Lin, Dekang. 1994. Principar - an efficient, broad-coverage, principle-based parser. In *COLING*. pages 482–488.

Lin, Dekang and Patrick Pantel. 2001. DIRT - discovery of inference rules from text. In *Knowledge Discovery and Data Mining*. pages 323–328.

Pedersen, T., S. Patwardhan, and J. Michelizzi. 2004. Wordnet::similarity - measuring the relatedness of concepts. In *Proc. of AAAI-04*.

Pereira, F., N. Tishby, and L. Lee. 1993. Distributional clustering of english words. In *Meeting of the Association for Computational Linguistics*. pages 183–190.

Resnik, Philip. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI*. pages 448–453.

Slonim, N., N. Friedman, and N. Tishby. 2002. Unsupervised document classification using sequential information maximization. In *Proc. of SIGIR'02*.

Slonim, N. and N. Tishby. 1999. Agglomerative information bottleneck. In *Proc. of NIPS-12*.

Torisawa, Kentaro. 2006. Acquiring inference rules with temporal constraints by using japanese coordinated sentences and noun-verb co-occurrences. In *Proceedings of NAACL*. pages 57–64.

Instance-based Evaluation of Entailment Rule Acquisition

Idan Szpektor, Eyal Shnarch, Ido Dagan
Dept. of Computer Science
Bar Ilan University
Ramat Gan, Israel
{szpekti,shey,dagan}@cs.biu.ac.il

Abstract

Obtaining large volumes of inference knowledge, such as entailment rules, has become a major factor in achieving robust semantic processing. While there has been substantial research on learning algorithms for such knowledge, their evaluation methodology has been problematic, hindering further research. We propose a novel evaluation methodology for entailment rules which explicitly addresses their semantic properties and yields satisfactory human agreement levels. The methodology is used to compare two state of the art learning algorithms, exposing critical issues for future progress.

1 Introduction

In many NLP applications, such as Question Answering (QA) and Information Extraction (IE), it is crucial to recognize that a particular target meaning can be inferred from different text variants. For example, a QA system needs to identify that "*Aspirin lowers the risk of heart attacks*" can be inferred from "*Aspirin prevents heart attacks*" in order to answer the question "*What lowers the risk of heart attacks?*". This type of reasoning has been recognized as a core semantic inference task by the generic *textual entailment* framework (Dagan et al., 2006).

A major obstacle for further progress in semantic inference is the lack of broad-scale knowledge-bases for semantic variability patterns (Bar-Haim et al., 2006). One prominent type of inference knowledge representation is inference rules such as paraphrases and *entailment rules*. We define an entailment rule to be a directional relation between two *templates*, text patterns with variables, e.g. 'X prevent $Y \rightarrow X$ lower the risk of Y'. The left-hand-side template is assumed to entail the right-hand-side template in certain contexts, under the same variable instantiation. Paraphrases can be viewed as bidirectional entailment rules. Such rules capture basic inferences and are used as building blocks for more complex entailment inference. For example, given the above rule, the answer "*Aspirin*" can be identified in the example above.

The need for large-scale inference knowledge-bases triggered extensive research on automatic acquisition of paraphrase and entailment rules. Yet the current precision of acquisition algorithms is typically still mediocre, as illustrated in Table 1 for DIRT (Lin and Pantel, 2001) and TEASE (Szpektor et al., 2004), two prominent acquisition algorithms whose outputs are publicly available. The current performance level only stresses the obvious need for satisfactory evaluation methodologies that would drive future research.

The prominent approach in the literature for evaluating rules, termed here the *rule-based* approach, is to present the rules to human judges asking whether each rule is correct or not. However, it is difficult to explicitly define when a learned rule should be considered correct under this methodology, and this was mainly left undefined in previous works. As the criterion for evaluating a rule is not well defined, using this approach often caused low agreement between human judges. Indeed, the standards for evaluation in this field are lower than other fields: many papers

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 456–463,
Prague, Czech Republic, June 2007.

don't report on human agreement at all and those that do report rather low agreement levels. Yet it is crucial to reliably assess rule correctness in order to measure and compare the performance of different algorithms in a replicable manner. Lacking a good evaluation methodology has become a barrier for further advances in the field.

In order to provide a well-defined evaluation methodology we first explicitly specify when entailment rules should be considered correct, following the spirit of their usage in applications. We then propose a new *instance-based* evaluation approach. Under this scheme, judges are not presented only with the rule but rather with a sample of sentences that match its left hand side. The judges then assess whether the rule holds under each specific example. A rule is considered correct only if the percentage of examples assessed as correct is sufficiently high.

We have experimented with a sample of input verbs for both DIRT and TEASE. Our results show significant improvement in human agreement over the rule-based approach. It is also the first comparison between such two state-of-the-art algorithms, which showed that they are comparable in precision but largely complementary in their coverage.

Additionally, the evaluation showed that both algorithms learn mostly one-directional rules rather than (symmetric) paraphrases. While most NLP applications need directional inference, previous acquisition works typically expected that the learned rules would be paraphrases. Under such an expectation, unidirectional rules were assessed as incorrect, underestimating the true potential of these algorithms. In addition, we observed that many learned rules are context sensitive, stressing the need to learn contextual constraints for rule applications.

2 Background: Entailment Rules and their Evaluation

2.1 Entailment Rules

An entailment rule '$L \rightarrow R$' is a directional relation between two templates, L and R. For example, 'X acquire $Y \rightarrow X$ own Y' or 'X beat $Y \rightarrow X$ play against Y'. Templates correspond to text fragments with variables, and are typically either linear phrases or parse sub-trees.

The goal of entailment rules is to help applications infer one text variant from another. A rule can be applied to a given text only when L can be inferred from it, with appropriate variable instantiation. Then, using the rule, the application deduces that R can also be inferred from the text under the same variable instantiation. For example, the rule 'X lose to $Y \rightarrow Y$ beat X' can be used to infer "*Liverpool beat Chelsea*" from "*Chelsea lost to Liverpool in the semifinals*".

Input	Correct	Incorrect
X change Y (DIRT)	($\leftrightarrow$) X modify Y ($\leftarrow$) X amend Y ($\leftarrow$) X revise Y	X adopt Y X create Y X stick to Y
X change Y (TEASE)	($\leftrightarrow$) X alter Y ($\rightarrow$) X affect Y ($\leftarrow$) X extend Y	X maintain Y X follow Y X use Y

Table 1: Examples of templates suggested by DIRT and TEASE as having an entailment relation, in some direction, with the input template 'X change Y'. The entailment direction arrows were judged manually and added for readability.

Entailment rules should typically be applied only in specific contexts, which we term *relevant contexts*. For example, the rule 'X acquire $Y \rightarrow X$ buy Y' can be used in the context of 'buying' events. However, it shouldn't be applied for "*Students acquired a new language*". In the same manner, the rule 'X acquire $Y \rightarrow X$ learn Y' should be applied only when Y corresponds to some sort of knowledge, as in the latter example.

Some existing entailment acquisition algorithms can add contextual constraints to the learned rules (Sekine, 2005), but most don't. However, NLP applications usually implicitly incorporate some contextual constraints when applying a rule. For example, when answering the question "*Which companies did IBM buy?*" a QA system would apply the rule 'X acquire $Y \rightarrow X$ buy Y' correctly, since the phrase "IBM acquire X" is likely to be found mostly in relevant economic contexts. We thus expect that an evaluation methodology should consider context relevance for entailment rules. For example, we would like both 'X acquire $Y \rightarrow X$ buy Y' and 'X acquire $Y \rightarrow X$ learn Y' to be assessed as correct (the second rule should not be deemed incorrect

just because it is not applicable in frequent economic contexts).

Finally, we highlight that the common notion of "paraphrase rules" can be viewed as a special case of entailment rules: a paraphrase '$L \leftrightarrow R$' holds if both templates entail each other. Following the textual entailment formulation, we observe that many applied inference settings require only directional entailment, and a requirement for symmetric paraphrase is usually unnecessary. For example, in order to answer the question "*Who owns Overture?*" it suffices to use a directional entailment rule whose right hand side is 'X own Y', such as 'X acquire $Y \rightarrow X$ own Y', which is clearly not a paraphrase.

2.2 Evaluation of Acquisition Algorithms

Many methods for automatic acquisition of rules have been suggested in recent years, ranging from distributional similarity to finding shared contexts (Lin and Pantel, 2001; Ravichandran and Hovy, 2002; Shinyama et al., 2002; Barzilay and Lee, 2003; Szpektor et al., 2004; Sekine, 2005). However, there is still no common accepted framework for their evaluation. Furthermore, all these methods learn rules as pairs of templates $\{L, R\}$ in a symmetric manner, without addressing rule directionality. Accordingly, previous works (except (Szpektor et al., 2004)) evaluated the learned rules under the paraphrase criterion, which underestimates the practical utility of the learned rules (see Section 2.1).

One approach which was used for evaluating automatically acquired rules is to measure their contribution to the performance of specific systems, such as QA (Ravichandran and Hovy, 2002) or IE (Sudo et al., 2003; Romano et al., 2006). While measuring the impact of learned rules on applications is highly important, it cannot serve as the primary approach for evaluating acquisition algorithms for several reasons. First, developers of acquisition algorithms often do not have access to the different applications that will later use the learned rules as generic modules. Second, the learned rules may affect individual systems differently, thus making observations that are based on different systems incomparable. Third, within a complex system it is difficult to assess the exact quality of entailment rules independently of effects of other system components.

Thus, as in many other NLP learning settings, a direct evaluation is needed. Indeed, the prominent approach for evaluating the quality of rule acquisition algorithms is by human judgment of the learned rules (Lin and Pantel, 2001; Shinyama et al., 2002; Barzilay and Lee, 2003; Pang et al., 2003; Szpektor et al., 2004; Sekine, 2005). In this evaluation scheme, termed here the *rule-based* approach, a sample of the learned rules is presented to the judges who evaluate whether each rule is correct or not. The criterion for correctness is not explicitly described in most previous works. By the common view of context relevance for rules (see Section 2.1), a rule was considered correct if the judge could think of reasonable contexts under which it holds.

We have replicated the rule-based methodology but did not manage to reach a 0.6 Kappa agreement level between pairs of judges. This approach turns out to be problematic because the rule correctness criterion is not sufficiently well defined and is hard to apply. While some rules might obviously be judged as correct or incorrect (see Table 1), judgment is often more difficult due to context relevance. One judge might come up with a certain context that, to her opinion, justifies the rule, while another judge might not imagine that context or think that it doesn't sufficiently support rule correctness. For example, in our experiments one of the judges did not identify the valid "religious holidays" context for the correct rule 'X observe $Y \rightarrow X$ celebrate Y'. Indeed, only few earlier works reported inter-judge agreement level, and those that did reported rather low Kappa values, such as 0.54 (Barzilay and Lee, 2003) and 0.55 - 0.63 (Szpektor et al., 2004).

To conclude, the prominent rule-based methodology for entailment rule evaluation is not sufficiently well defined. It results in low inter-judge agreement which prevents reliable and consistent assessments of different algorithms.

3 Instance-based Evaluation Methodology

As discussed in Section 2.1, an evaluation methodology for entailment rules should reflect the expected validity of their application within NLP systems. Following that line, an entailment rule '$L \rightarrow R$' should be regarded as *correct* if in all (or at least most) relevant contexts in which the instantiated template L is inferred from the given text, the instan-

	Rule	Sentence	Judgment
1	X seek $Y \rightarrow X$ disclose Y	If he is arrested, **he** can immediately seek **bail**.	Left not entailed
2	X clarify $Y \rightarrow X$ prepare Y	**He** didn't clarify **his position on the subject**.	Left not entailed
3	X hit $Y \rightarrow X$ approach Y	**Other earthquakes** have hit **Lebanon** since '82.	Irrelevant context
4	X lose $Y \rightarrow X$ surrender Y	**Bread** has recently lost **its subsidy**.	Irrelevant context
5	X regulate $Y \rightarrow X$ reform Y	**The SRA** regulates **the sale of sugar**.	No entailment
6	X resign $Y \rightarrow X$ share Y	**Lopez** resigned **his post at VW** last week.	No entailment
7	X set $Y \rightarrow X$ allow Y	**The committee** set **the following refunds**.	Entailment holds
8	X stress $Y \rightarrow X$ state Y	**Ben Yahia** also stressed **the need for action**.	Entailment holds

Table 2: Rule evaluation examples and their judgment.

tiated template R is also inferred from the text. This reasoning corresponds to the common definition of entailment in semantics, which specifies that a text L entails another text R if R is true in every circumstance (possible world) in which L is true (Chierchia and McConnell-Ginet, 2000).

It follows that in order to assess if a rule is correct we should judge whether R is typically entailed from those sentences that entail L (within relevant contexts for the rule). We thus present a new evaluation scheme for entailment rules, termed the *instance-based* approach. At the heart of this approach, human judges are presented not only with a rule but rather with a sample of examples of the rule's usage. Instead of thinking up valid contexts for the rule the judges need to assess the rule's validity under the given context in each example. The essence of our proposal is a (apparently non-trivial) protocol of a sequence of questions, which determines rule validity in a given sentence.

We shall next describe how we collect a sample of examples for evaluation and the evaluation process.

3.1 Sampling Examples

Given a rule '$L \rightarrow R$', our goal is to generate evaluation examples by finding a sample of sentences from which L is entailed. We do that by automatically retrieving, from a given corpus, sentences that match L and are thus likely to entail it, as explained below.

For each example sentence, we automatically extract the arguments that instantiate L and generate two phrases, termed *left phrase* and *right phrase*, which are constructed by instantiating the left template L and the right template R with the extracted arguments. For example, the left and right phrases generated for example 1 in Table 2 are "*he seek bail*" and "*he disclose bail*", respectively.

Finding sentences that match L can be performed at different levels. In this paper we match lexical-syntactic templates by finding a sub-tree of the sentence parse that is identical to the template structure. Of course, this matching method is not perfect and will sometimes retrieve sentences that do not entail the left phrase for various reasons, such as incorrect sentence analysis or semantic aspects like negation, modality and conditionals. See examples 1-2 in Table 2 for sentences that syntactically match L but do not entail the instantiated left phrase. Since we should assess R's entailment only from sentences that entail L, such sentences should be ignored by the evaluation process.

3.2 Judgment Questions

For each example generated for a rule, the judges are presented with the given sentence and the left and right phrases. They primarily answer two questions that assess whether entailment holds in this example, following the semantics of entailment rule application as discussed above:

$\mathbf{Q_{le}}$: Is the left phrase entailed from the sentence? A positive/negative answer corresponds to a '**Left entailed/not entailed**' judgment.

$\mathbf{Q_{re}}$: Is the right phrase entailed from the sentence? A positive/negative answer corresponds to an '**Entailment holds/No entailment**' judgment.

The first question identifies sentences that do not entail the left phrase, and thus should be ignored when evaluating the rule's correctness. While inappropriate matches of the rule left-hand-side may happen

and harm an overall system precision, such errors should be accounted for a system's rule matching module rather than for the rules' precision. The second question assesses whether the rule application is valid or not for the current example. See examples 5-8 in Table 2 for cases where entailment does or doesn't hold.

Thus, the judges focus only on the given sentence in each example, so the task is actually to evaluate whether *textual entailment* holds between the sentence (*text*) and each of the left and right phrases (*hypotheses*). Following past experience in textual entailment evaluation (Dagan et al., 2006) we expect a reasonable agreement level between judges.

As discussed in Section 2.1, we may want to ignore examples whose context is irrelevant for the rule. To optionally capture this distinction, the judges are asked another question:

$\mathbf{Q_{rc}}$: Is the right phrase a likely phrase in English? A positive/negative answer corresponds to a '**Relevant/Irrelevant context**' evaluation.

If the right phrase is not likely in English then the given context is probably irrelevant for the rule, because it seems inherently incorrect to infer an implausible phrase. Examples 3-4 in Table 2 demonstrate cases of irrelevant contexts, which we may choose to ignore when assessing rule correctness.

3.3 Evaluation Process

For each example, the judges are presented with the three questions above in the following order: (1) $\mathbf{Q_{le}}$ (2) $\mathbf{Q_{rc}}$ (3) $\mathbf{Q_{re}}$. If the answer to a certain question is negative then we do not need to present the next questions to the judge: if the left phrase is not entailed then we ignore the sentence altogether; and if the context is irrelevant then the right phrase cannot be entailed from the sentence and so the answer to $\mathbf{Q_{re}}$ is already known as negative.

The above entailment judgments assume that we can actually ask whether the left or right phrases are correct given the sentence, that is, we assume that a truth value can be assigned to both phrases. This is the case when the left and right templates correspond, as expected, to semantic relations. Yet sometimes learned templates are (erroneously) not relational, e.g. 'X, Y, IBM' (representing a list). We therefore let the judges initially mark rules that include such templates as non-relational, in which case their examples are not evaluated at all.

3.4 Rule Precision

We compute the precision of a rule by the percentage of examples for which entailment holds out of all "relevant" examples. We can calculate the precision in two ways, as defined below, depending on whether we ignore irrelevant contexts or not (obtaining lower precision if we don't). When systems answer an information need, such as a query or question, irrelevant contexts are sometimes not encountered thanks to additional context which is present in the given input (see Section 2.1). Thus, the following two measures can be viewed as upper and lower bounds for the expected precision of the rule applications in actual systems:

upper bound precision: $$\frac{\#\textbf{Entailment holds}}{\#\textbf{Relevant context}}$$

lower bound precision: $$\frac{\#\textbf{Entailment holds}}{\#\textbf{Left entailed}}$$

where # denotes the number of examples with the corresponding judgment.

Finally, we consider a rule to be correct only if its precision is at least 80%, which seems sensible for typical applied settings. This yields two alternative sets of correct rules, corresponding to the upper bound and lower bound precision measures. Even though judges may disagree on specific examples for a rule, their judgments may still agree overall on the rule's correctness. We therefore expect the agreement level on rule correctness to be higher than the agreement on individual examples.

4 Experimental Settings

We applied the instance-based methodology to evaluate two state-of-the-art unsupervised acquisition algorithms, DIRT (Lin and Pantel, 2001) and TEASE (Szpektor et al., 2004), whose output is publicly available. DIRT identifies semantically related templates in a local corpus using distributional similarity over the templates' variable instantiations. TEASE acquires entailment relations from the Web for a given input template I by identifying characteristic variable instantiations shared by I and other templates.

For the experiment we used the published DIRT and TEASE knowledge-bases[1]. For every given input template I, each knowledge-base provides a list of learned output templates $\{O_j\}_1^{n_I}$, where n_I is the number of output templates learned for I. Each output template is suggested as holding an entailment relation with the input template I, but the algorithms do not specify the entailment direction(s). Thus, each pair $\{I, O_j\}$ induces two candidate directional entailment rules: '$I \rightarrow O_j$' and '$O_j \rightarrow I$'.

4.1 Test Set Construction

The test set construction consists of three sampling steps: selecting a set of input templates for the two algorithms, selecting a sample of output rules to be evaluated, and selecting a sample of sentences to be judged for each rule.

First, we randomly selected 30 transitive verbs out of the 1000 most frequent verbs in the Reuters RCV1 corpus[2]. For each verb we manually constructed a lexical-syntactic input template by adding subject and object variables. For example, for the verb 'seek' we constructed the template '$X \xleftarrow{subj}$ seek $\xrightarrow{obj} Y$'.

Next, for each input template I we considered the learned templates $\{O_j\}_1^{n_I}$ from each knowledge-base. Since DIRT has a long tail of templates with a low score and very low precision, DIRT templates whose score is below a threshold of 0.1 were filtered out[3]. We then sampled 10% of the templates in each output list, limiting the sample size to be between 5-20 templates for each list (thus balancing between sufficient evaluation data and judgment load). For each sampled template O we evaluated both directional rules, '$I \rightarrow O$' and '$O \rightarrow I$'. In total, we sampled 380 templates, inducing 760 directional rules out of which 754 rules were unique.

Last, we randomly extracted a sample of example sentences for each rule '$L \rightarrow R$' by utilizing a search engine over the first CD of Reuters RCV1. First, we retrieved all sentences containing all lexical terms within L. The retrieved sentences were parsed using the Minipar dependency parser (Lin, 1998), keeping only sentences that syntactically match L (as explained in Section 3.1). A sample of 15 matching sentences was randomly selected, or all matching sentences if less than 15 were found. Finally, an example for judgment was generated from each sampled sentence and its left and right phrases (see Section 3.1). We did not find sentences for 108 rules, and thus we ended up with 646 unique rules that could be evaluated (with 8945 examples to be judged).

4.2 Evaluating the Test-Set

Two human judges evaluated the examples. We randomly split the examples between the judges. 100 rules (1287 examples) were cross annotated for agreement measurement. The judges followed the procedure in Section 3.3 and the correctness of each rule was assessed based on both its upper and lower bound precision values (Section 3.4).

5 Methodology Evaluation Results

We assessed the instance-based methodology by measuring the agreement level between judges. The judges agreed on 75% of the 1287 shared examples, corresponding to a reasonable Kappa value of 0.64. A similar kappa value of 0.65 was obtained for the examples that were judged as either entailment holds/no entailment by both judges. Yet, our evaluation target is to assess rules, and the Kappa values for the final correctness judgments of the shared rules were 0.74 and 0.68 for the lower and upper bound evaluations. These Kappa scores are regarded as 'substantial agreement' and are substantially higher than published agreement scores and those we managed to obtain using the standard rule-based approach. As expected, the agreement on rules is higher than on examples, since judges may disagree on a certain example but their judgements would still yield the same rule assessment.

Table 3 illustrates some disagreements that were still exhibited within the instance-based evaluation. The primary reason for disagreements was the difficulty to decide whether a context is relevant for a rule or not, resulting in some confusion between 'Irrelevant context' and 'No entailment'. This may explain the lower agreement for the upper bound precision, for which examples judged as 'Irrelevant context' are ignored, while for the lower bound both

[1] Available at http://aclweb.org/aclwiki/index.php?title=Textual_Entailment_Resource_Pool

[2] http://about.reuters.com/researchandstandards/corpus/

[3] Following advice by Patrick Pantel, DIRT's co-author.

Rule	Sentence	Judge 1	Judge 2
X sign $Y \rightarrow X$ set Y	**Iraq and Turkey** sign **agreement** to increase trade cooperation	Entailment holds	Irrelevant context
X worsen $Y \rightarrow X$ slow Y	**News of the strike** worsened **the situation**	Irrelevant context	No entailment
X get $Y \rightarrow X$ want Y	**He** will get **his parade** on Tuesday	Entailment holds	No entailment

Table 3: Examples for disagreement between the two judges.

judgments are conflated and represent no entailment. Our findings suggest that better ways for distinguishing relevant contexts may be sought in future research for further refinement of the instance-based evaluation methodology.

About 43% of all examples were judged as 'Left not entailed'. The relatively low matching precision (57%) made us collect more examples than needed, since 'Left not entailed' examples are ignored. Better matching capabilities will allow collecting and judging fewer examples, thus improving the efficiency of the evaluation process.

6 DIRT and TEASE Evaluation Results

	DIRT		TEASE	
	P	Y	P	Y
Rules:				
Upper Bound	30.5%	33.5	28.4%	40.3
Lower Bound	18.6%	20.4	17%	24.1
Templates:				
Upper Bound	44%	22.6	38%	26.9
Lower Bound	27.3%	14.1	23.6%	16.8

Table 4: Average Precision (P) and Yield (Y) at the rule and template levels.

We evaluated the quality of the entailment rules produced by each algorithm using two scores: (1) micro average ***Precision***, the percentage of correct rules out of all learned rules, and (2) average ***Yield***, the average number of correct rules learned for each input template I, as extrapolated based on the sample[4]. Since DIRT and TEASE do not identify rule directionality, we also measured these scores at the template level, where an output template O is considered correct if at least one of the rules '$I \rightarrow O$' or '$O \rightarrow I$' is correct. The results are presented in Table 4. The major finding is that the overall quality of DIRT and TEASE is very similar. Under the specific DIRT cutoff threshold chosen, DIRT exhibits somewhat higher Precision while TEASE has somewhat higher Yield (recall that there is no particular natural cutoff point for DIRT's output).

Since applications typically apply rules in a specific direction, the Precision for rules reflects their expected performance better than the Precision for templates. Obviously, future improvement in precision is needed for rule learning algorithms. Meanwhile, manual filtering of the learned rules can prove effective within limited domains, where our evaluation approach can be utilized for reliable filtering as well. The substantial yield obtained by these algorithms suggest that they are indeed likely to be valuable for recall increase in semantic applications.

In addition, we found that only about 15% of the correct templates were learned by both algorithms, which implies that the two algorithms largely complement each other in terms of coverage. One explanation may be that DIRT is focused on the domain of the local corpus used (news articles for the published DIRT knowledge-base), whereas TEASE learns from the Web, extracting rules from multiple domains. Since Precision is comparable it may be best to use both algorithms in tandem.

We also measured whether O is a paraphrase of I, i.e. whether both '$I \rightarrow O$' and '$O \rightarrow I$' are correct. Only 20-25% of all correct templates were assessed as paraphrases. This stresses the significance of evaluating directional rules rather than only paraphrases. Furthermore, it shows that in order to improve precision, acquisition algorithms must identify rule directionality.

[4]Since the rules are matched against the full corpus (as in IR evaluations), it is difficult to evaluate their true recall.

About 28% of all 'Left entailed' examples were evaluated as 'Irrelevant context', yielding the large difference in precision between the upper and lower precision bounds. This result shows that in order to get closer to the upper bound precision, learning algorithms and applications need to identify the relevant contexts in which a rule should be applied.

Last, we note that the instance-based quality assessment corresponds to the corpus from which the example sentences were taken. It is therefore best to evaluate the rules using a corpus of the same domain from which they were learned, or the target application domain for which the rules will be applied.

7 Conclusions

Accurate learning of inference knowledge, such as entailment rules, has become critical for further progress of applied semantic systems. However, evaluation of such knowledge has been problematic, hindering further developments. The instance-based evaluation approach proposed in this paper obtained acceptable agreement levels, which are substantially higher than those obtained for the common rule-based approach.

We also conducted the first comparison between two state-of-the-art acquisition algorithms, DIRT and TEASE, using the new methodology. We found that their quality is comparable but they effectively complement each other in terms of rule coverage. Also, we found that most learned rules are not paraphrases but rather one-directional entailment rules, and that many of the rules are context sensitive. These findings suggest interesting directions for future research, in particular learning rule directionality and relevant contexts, issues that were hardly explored till now. Such developments can be then evaluated by the instance-based methodology, which was designed to capture these two important aspects of entailment rules.

Acknowledgements

The authors would like to thank Ephi Sachs and Iddo Greental for their evaluation. This work was partially supported by ISF grant 1095/05, the IST Programme of the European Community under the PASCAL Network of Excellence IST-2002-506778, and the ITC-irst/University of Haifa collaboration.

References

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Second PASCAL Challenge Workshop for Recognizing Textual Entailment*.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of NAACL-HLT*.

Gennaro Chierchia and Sally McConnell-Ginet. 2000. *Meaning and Grammar (2nd ed.): an introduction to semantics*. MIT Press, Cambridge, MA.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. *Lecture Notes in Computer Science*, 3944:177–190.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360.

Dekang Lin. 1998. Dependency-based evaluation of minipar. In *Proceedings of the Workshop on Evaluation of Parsing Systems at LREC*.

Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of HLT-NAACL*.

Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of ACL*.

Lorenza Romano, Milen Kouylekov, Idan Szpektor, Ido Dagan, and Alberto Lavelli. 2006. Investigating a generic paraphrase-based approach for relation extraction. In *Proceedings of EACL*.

Satoshi Sekine. 2005. Automatic paraphrase discovery based on context and keywords between ne pairs. In *Proceedings of IWP*.

Yusuke Shinyama, Satoshi Sekine, Kiyoshi Sudo, and Ralph Grishman. 2002. Automatic paraphrase acquisition from news articles. In *Proceedings of HLT*.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An improved extraction pattern representation model for automatic IE pattern acquisition. In *Proceedings of ACL*.

Idan Szpektor, Hristo Tanev, Ido Dagan, and Bonaventura Coppola. 2004. Scaling web-based acquisition of entailment relations. In *Proceedings of EMNLP*.

Statistical Machine Translation for Query Expansion in Answer Retrieval

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal and **Yi Liu**
Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA 94043
{riezler|avasserm|ioannis|vibhu|yliu}@google.com

Abstract

We present an approach to query expansion in answer retrieval that uses Statistical Machine Translation (SMT) techniques to bridge the lexical gap between questions and answers. SMT-based query expansion is done by i) using a full-sentence paraphraser to introduce synonyms in context of the entire query, and ii) by translating query terms into answer terms using a full-sentence SMT model trained on question-answer pairs. We evaluate these global, context-aware query expansion techniques on *tfidf* retrieval from 10 million question-answer pairs extracted from FAQ pages. Experimental results show that SMT-based expansion improves retrieval performance over local expansion and over retrieval without expansion.

1 Introduction

One of the fundamental problems in Question Answering (QA) has been recognized to be the "lexical chasm" (Berger et al., 2000) between question strings and answer strings. This problem is manifested in a mismatch between question and answer vocabularies, and is aggravated by the inherent ambiguity of natural language. Several approaches have been presented that apply natural language processing technology to close this gap. For example, syntactic information has been deployed to reformulate questions (Hermjakob et al., 2002) or to replace questions by syntactically similar ones (Lin and Pantel, 2001); lexical ontologies such as Wordnet[1] have been used to find synonyms for question words (Burke et al., 1997; Hovy et al., 2000; Prager et al., 2001; Harabagiu et al., 2001), and statistical machine translation (SMT) models trained on question-answer pairs have been used to rank candidate answers according to their translation probabilities (Berger et al., 2000; Echihabi and Marcu, 2003; Soricut and Brill, 2006). Information retrieval (IR) is faced by a similar fundamental problem of "term mismatch" between queries and documents. A standard IR solution, query expansion, attempts to increase the chances of matching words in relevant documents by adding terms with similar statistical properties to those in the original query (Voorhees, 1994; Qiu and Frei, 1993; Xu and Croft, 1996).

In this paper we will concentrate on the task of answer retrieval from FAQ pages, i.e., an IR problem where user queries are matched against documents consisting of question-answer pairs found in FAQ pages. Equivalently, this is a QA problem that concentrates on finding answers given FAQ documents that are known to contain the answers. Our approach to close the lexical gap in this setting attempts to marry QA and IR technology by deploying SMT methods for query expansion in answer retrieval. We present two approaches to SMT-based query expansion, both of which are implemented in the framework of phrase-based SMT (Och and Ney, 2004; Koehn et al., 2003).

Our first query expansion model trains an end-to-end phrase-based SMT model on 10 million question-answer pairs extracted from FAQ pages.

[1] http://wordnet.princeton.edu

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 464–471,
Prague, Czech Republic, June 2007.

The goal of this system is to learn lexical correlations between words and phrases in questions and answers, for example by allowing for multiple unaligned words in automatic word alignment, and disregarding issues such as word order. The ability to translate phrases instead of words and the use of a large language model serve as rich context to make precise decisions in the case of ambiguous translations. Query expansion is performed by adding content words that have not been seen in the original query from the n-best translations of the query.

Our second query expansion model is based on the use of SMT technology for full-sentence paraphrasing. A phrase table of paraphrases is extracted from bilingual phrase tables (Bannard and Callison-Burch, 2005), and paraphrasing quality is improved by additional discriminative training on manually created paraphrases. This approach utilizes large bilingual phrase tables as information source to extract a table of para-phrases. Synonyms for query expansion are read off from the n-best paraphrases of full queries instead of from paraphrases of separate words or phrases. This allows the model to take advantage of the rich context of a large n-gram language model when adding terms from the n-best paraphrases to the original query.

In our experimental evaluation we deploy a database of question-answer pairs extracted from FAQ pages for both training a question-answer translation model, and for a comparative evaluation of different systems on the task of answer retrieval. Retrieval is based on the *tfidf* framework of Jijkoun and de Rijke (2005), and query expansion is done straightforwardly by adding expansion terms to the query for a second retrieval cycle. We compare our global, context-aware query expansion techniques with Jijkoun and de Rijke's (2005) *tfidf* model for answer retrieval and a local query expansion technique (Xu and Croft, 1996). Experimental results show a significant improvement of SMT-based query expansion over both baselines.

2 Related Work

QA has approached the problem of the lexical gap by various techniques for *question reformulation*, including rule-based syntactic and semantic reformulation patterns (Hermjakob et al., 2002), reformulations based on shared dependency parses (Lin and Pantel, 2001), or various uses of the WordNet ontology to close the lexical gap word-by-word (Hovy et al., 2000; Prager et al., 2001; Harabagiu et al., 2001). Another use of natural language processing has been the deployment of SMT models on question-answer pairs for *(re)ranking* candidate answers which were either assumed to be contained in FAQ pages (Berger et al., 2000) or retrieved by baseline systems (Echihabi and Marcu, 2003; Soricut and Brill, 2006).

IR has approached the term mismatch problem by various approaches to *query expansion* (Voorhees, 1994; Qiu and Frei, 1993; Xu and Croft, 1996). Inconclusive results have been reported for techniques that expand query terms separately by adding strongly related terms from an external thesaurus such as WordNet (Voorhees, 1994). Significant improvements in retrieval performance could be achieved by *global* expansion techniques that compute corpus-wide statistics and take the entire query, or query *concept* (Qiu and Frei, 1993), into account, or by *local* expansion techniques that select expansion terms from the top ranked documents retrieved by the original query (Xu and Croft, 1996).

A similar picture emerges for query expansion in QA: Mixed results have been reported for word-by-word expansion based on WordNet (Burke et al., 1997; Hovy et al., 2000; Prager et al., 2001; Harabagiu et al., 2001). Considerable improvements have been reported for the use of the local context analysis model of Xu and Croft (1996) in the QA system of Ittycheriah et al. (2001), or for the systems of Agichtein et al. (2004) or Harabagiu and Lacatusu (2004) that use FAQ data to learn how to expand query terms by answer terms.

The SMT-based approaches presented in this paper can be seen as global query expansion techniques in that our question-answer translation model uses the whole question-answer corpus as information source, and our approach to paraphrasing deploys large amounts of bilingual phrases as high-coverage information source for synonym finding. Furthermore, both approaches take the entire query context into account when proposing to add new terms to the original query. The approaches that are closest to our models are the SMT approach of Radev et al. (2001) and the paraphrasing approach

	web pages	FAQ pages	QA pairs
count	4 billion	795,483	10,568,160

Table 1: Corpus statistics of QA pair data

of Duboue and Chu-Carroll (2006). None of these approaches defines the problem of the lexical gap as a query expansion problem, and both approaches use much simpler SMT models than our systems, e.g., Radev et al. (2001) neglect to use a language model to aid disambiguation of translation choices, and Duboue and Chu-Carroll (2006) use SMT as black box altogether.

In sum, our approach differs from previous work in QA and IR in the use SMT technology for query expansion, and should be applicable in both areas even though experimental results are only given for the restricted domain of retrieval from FAQ pages.

3 Question-Answer Pairs from FAQ Pages

Large-scale collection of question-answer pairs has been hampered in previous work by the small sizes of publicly available FAQ collections or by restricted access to retrieval results via public APIs of search engines. Jijkoun and de Rijke (2005) nevertheless managed to extract around 300,000 FAQ pages and 2.8 million question-answer pairs by repeatedly querying search engines with "`intitle:faq`" and "`inurl:faq`". Soricut and Brill (2006) could deploy a proprietary URL collection of 1 billion URLs to extract 2.3 million FAQ pages containing the uncased string "faq" in the url string. The extraction of question-answer pairs amounted to a database of 1 million pairs in their experiment. However, inspection of the publicly available WebFAQ collection provided by Jijkoun and de Rijke[2] showed a great amount of noise in the retrieved FAQ pages and question-answer pairs, and yet the indexed question-answer pairs showed a serious recall problem in that no answer could be retrieved for many well-formed queries. For our experiment, we decided to prefer precision over recall and to attempt a precision-oriented FAQ and question-answer pair extraction that benefits the training of question-answer translation models.

[2]http://ilps.science.uva.nl/Resources/WazDah/

As shown in Table 1, the FAQ pages used in our experiment were extracted from a 4 billion page subset of the web using the queries "`inurl:faq`" and "`inurl:faqs`" to match the tokens "faq" or "faqs" in the urls. This extraction resulted in 2.6 million web pages (0.07% of the crawl). Since not all those pages are actually FAQs, we manually labeled 1,000 of those pages to train an online passive-aggressive classificier (Crammer et al., 2006) in a 10-fold cross validation setup. Training was done using 20 feature functions on occurrences question marks and key words in different fields of web pages, and resulted in an F1 score of around 90% for FAQ classification. Application of the classifier to the extracted web pages resulted in a classification of 795,483 pages as FAQ pages.

The extraction of question-answer pairs from this database of FAQ pages was performed again in a precision-oriented manner. The goal of this step was to extract url, title, question, and answers fields from the question-answer pairs in FAQ pages. This was achieved by using feature functions on punctuations, HTML tags (e.g., `<p>, <BR>`), listing markers (e.g., `Q:, (1)`), and lexical cues (e.g., `What, How`), and an algorithm similar to Joachims (2003) to propagate initial labels across similar text pieces. The result of this extraction step is a database of about 10 million question answer pairs (13.3 pairs per FAQ page). A manual evaluation of 100 documents, containing 1,303 question-answer pairs, achieved a precision of 98% and a recall of 82% for extracting question-answer pairs.

4 SMT-Based Query Expansion

Our SMT-based query expansion techniques are based on a recent implementation of the phrase-based SMT framework (Koehn et al., 2003; Och and Ney, 2004). The probability of translating a foreign sentence **f** into English **e** is defined in the noisy channel model as

$$\arg\max_e p(\mathbf{e}|\mathbf{f}) \quad = \quad \arg\max_e p(\mathbf{f}|\mathbf{e})p(\mathbf{e}) \quad (1)$$

This allows for a separation of a language model $p(\mathbf{e})$, and a translation model $p(\mathbf{f}|\mathbf{e})$. Translation probabilities are calculated from relative frequencies of phrases, which are extracted via various heuristics as larger blocks of aligned words from best word

alignments. Word alignments are estimated by models similar to Brown et al. (1993). For a sequence of I phrases, the translation probability in equation (1) can be decomposed into

$$p(f_i^I|e_i^I) = \prod_{i=1}^{I} p(f_i|e_i) \quad (2)$$

Recent SMT models have shown significant improvements in translation quality by improved modeling of local word order and idiomatic expressions through the use of phrases, and by the deployment of large n-gram language models to model fluency and lexical choice.

4.1 Question-Answer Translation

Our first approach to query expansion treats the questions and answers in the question-answer corpus as two distinct languages. That is, the 10 million question-answer pairs extracted from FAQ pages are fed as parallel training data into an SMT training pipeline. This training procedure includes various standard procedures such as preprocessing, sentence and chunk alignment, word alignment, and phrase extraction. The goal of question-answer translation is to learn associations between question words and synonymous answer words, rather than the translation of questions into fluent answers. Thus we did not conduct discriminative training of feature weights for translation probabilities or language model probabilities, but we held out 4,000 question-answer pairs for manual development and testing of the system. For example, the system was adjusted to account for the difference in sentence length between questions and answers by setting the null-word probability parameter in word alignment to 0.9. This allowed us to concentrate the word alignments to a small number of key words. Furthermore, extraction of phrases was based on the intersection of alignments from both translation directions, thus favoring precision over recall also in phrase alignment.

Table 2 shows unique translations of the query "how to live with cat allergies" on the phrase-level, with corresponding source and target phrases shown in brackets. Expansion terms are taken from phrase terms that have not been seen in the original query, and are highlighted in bold face.

4.2 SMT-Based Paraphrasing

Our SMT-based paraphrasing system is based on the approach presented in Bannard and Callison-Burch (2005). The central idea in this approach is to identify paraphrases or synonyms at the phrase level by pivoting on another language. For example, given a table of Chinese-to-English phrase translations, phrasal synonyms in the target language are defined as those English phrases that are aligned to the same Chinese source phrases. Translation probabilities for extracted para-phrases can be inferred from bilingual translation probabilities as follows: Given an English para-phrase pair (trg, syn), the probability $p(syn|trg)$ that trg translates into syn is defined as the joint probability that the English phrase trg translates into the foreign phrase src, and that the foreign phrase src translates into the English phrase syn. Under an independence assumption of those two events, this probability and the reverse translation direction $p(trg|syn)$ can be defined as follows:

$$p(syn|trg) = \max_{src} p(src|trg)p(syn|src) \quad (3)$$
$$p(trg|syn) = \max_{src} p(src|syn)p(trg|src)$$

Since the same para-phrase pair can be obtained by pivoting on multiple foreign language phrases, a summation or maximization over foreign language phrases is necessary. In order not to put too much probability mass onto para-phrase translations that can be obtained from multiple foreign language phrases, we maximize instead of summing over src.

In our experiments, we employed equation (3) to infer for each para-phrase pair translation model probabilities $p_\phi(syn|trg)$ and $p_{\phi'}(trg|syn)$ from relative frequencies of phrases in bilingual tables. In contrast to Bannard and Callison-Burch (2005), we applied the same inference step to infer also lexical translation probabilities $p_w(syn|trg)$ and $p_{w'}(trg|syn)$ as defined in Koehn et al. (2003) for para-phrases. Furthermore, we deployed features for the number of words l_w, number of phrases c_ϕ , a reordering score p_d , and a score for a 6-gram language model p_{LM} trained on English web data. The final model combines these features in a log-linear model that defines the probability of paraphrasing a full sentence, consisting of a sequence of I phrases

qa-translation	(how, how) (to, to) (live, live) (with, with) (cat, **pet**) (allergies, allergies) (how, how) (to, to) (live, live) (with, with) (cat, cat) (allergies, **allergy**) (how, how) (to, to) (live, live) (with, with) (cat, cat) (allergies, **food**) (how, how) (to, to) (live, live) (with, with) (cat, **cats**) (allergies, allergies)
paraphrasing	(how, how) (to live, to live) (with cat, with cat) (allergies, **allergy**) (how, **ways**) (to live, to live) (with cat, with cat) (allergies, allergies) (how, how) (to live with, to live with) (cat, **feline**) (allergies, allergies) (how to, how to) (live, **living**) (with cat, with cat) (allergies, allergies) (how to, how to) (live, **life**) (with cat, with cat) (allergies, allergies) (how, **way**) (to live, to live) (with cat, with cat) (allergies, allergies) (how, how) (to live, to live) (with cat, with cat) (allergies, **allergens**) (how, how) (to live, to live) (with cat, with cat) (allergies, **allergen**)

Table 2: Unique n-best phrase-level translations of query "how to live with cat allergies".

as follows:

$$
\begin{aligned}
p(syn_1^I|trg_1^I) &= (\prod_{i=1}^{I} p_\phi(syn_i|trg_i)^{\lambda_\phi} \quad (4)\\
&\times\ p_{\phi'}(trg_i|syn_i)^{\lambda_{\phi'}}\\
&\times\ p_w(syn_i|trg_i)^{\lambda_w}\\
&\times\ p_{w'}(trg_i|syn_i)^{\lambda_{w'}}\\
&\times\ p_d(syn_i, trg_i)^{\lambda_d})\\
&\times\ l_w(syn_1^I)^{\lambda_l}\\
&\times\ c_\phi(syn_1^I)^{\lambda_c}\\
&\times\ p_{LM}(syn_1^I)^{\lambda_{LM}}
\end{aligned}
$$

For estimation of the feature weights $\vec{\lambda}$ defined in equation (4) we employed minimum error rate (MER) training under the BLEU measure (Och, 2003). Training data for MER training were taken from multiple manual English translations of Chinese sources from the NIST 2006 evaluation data. The first of four reference translations for each Chinese sentence was taken as source paraphrase, the rest as reference paraphrases. Discriminative training was conducted on 1,820 sentences; final evaluation on 2,390 sentences. A baseline paraphrase table consisting of 33 million English para-phrase pairs was extracted from 1 billion phrase pairs from three different languages, at a cutoff of para-phrase probabilities of 0.0025.

Query expansion is done by adding terms introduced in n-best paraphrases of the query. Table 2 shows example paraphrases for the query "how to live with cat allergies" with newly introduced terms highlighted in bold face.

5 Experimental Evaluation

Our baseline answer retrieval system is modeled after the *tfidf* retrieval model of Jijkoun and de Rijke (2005). Their model calculates a linear combination of vector similarity scores between the user query and several fields in the question-answer pair. We used the cosine similarity metric with logarithmically weighted term and document frequency weights in order to reproduce the Lucene[3] model used in Jijkoun and de Rijke (2005). For indexing of fields, we adopted the settings that were reported to be optimal in Jijkoun and de Rijke (2005). These settings comprise the use of 8 question-answer pair fields, and a weight vector $\langle 0.0, 1.0, 0.0, 0.0, 0.5, 0.5, 0.2, 0.3\rangle$ for fields ordered as follows: (1) full FAQ document text, (2) question text, (3) answer text, (4) title text, (5)-(8) each of the above without stopwords. The second field thus takes takes wh-words, which would typically be filtered out, into account. All other fields are matched without stopwords, with higher weight assigned to document and question than to answer and title fields. We did not use phrase-matching or stemming in our experiments, similar to Jijkoun and de Rijke (2005), who could not find positive effects for these features in their experiments.

Expansion terms are taken from those terms in the n-best translations of the query that have not been seen in the original query string. For paraphrasing-based query expansion, a 50-best list of paraphrases of the original query was used. For the noisier question-answer translation, expansion terms and phrases were extracted from a 10-

[3] http://lucene.apache.org

	S_2@10	S_2@20	$S_{1,2}$@10	$S_{1,2}$@20
baseline *tfidf*	27	35	58	65
local expansion	30 (+ 11.1)	40 (+ 14.2)	57 (- 1)	63 (- 3)
SMT-based expansion	38 (+ 40.7)	43 (+ 22.8)	58	65

Table 3: Success rate at 10 or 20 results for retrieval of adequate (2) or material (1) answers; relative change in brackets.

best list of query translations. Terms taken from query paraphrases were matched with the same field weight vector $\langle 0.0, 1.0, 0.0, 0.0, 0.5, 0.5, 0.2, 0.3\rangle$ as above. Terms taken from question-answer translation were matched with the weight vector $\langle 0.0, 1.0, 0.0, 0.0, 0.5, 0.2, 0.5, 0.3\rangle$, preferring answer fields over question fields. After stopword removal, the average number of expansion terms produced was 7.8 for paraphrasing, and 3.1 for question-answer translation.

The local expansion technique used in our experiments follows Xu and Croft (1996) in taking expansion terms from the top n answers that were retrieved by the baseline *tfidf* system, and by incorporating cooccurrence information with query terms. This is done by calculating term frequencies for expansion terms by summing up the *tfidf* weights of the answers in which they occur, thus giving higher weight to terms that occur in answers that receive a higher similarity score to the original query. In our experiments, expansion terms are ranked according to this modified *tfidf* calculation over the top 20 answers retrieved by the baseline retrieval run, and matched a second time with the field weight vector $\langle 0.0, 1.0, 0.0, 0.0, 0.5, 0.2, 0.5, 0.3\rangle$ that prefers answer fields over question fields. After stopword removal, the average number of expansion terms produced by the local expansion technique was 9.25.

The test queries we used for retrieval are taken from query logs of the MetaCrawler search engine[4] and were provided to us by Valentin Jijkoun. In order to maximize recall for the comparative evaluation of systems, we selected 60 queries that were well-formed natural language questions without metacharacters and spelling errors. However, for one third of these well-formed queries none of the five compared systems could retrieve an answer. Examples are *"how do you make a cornhusk doll"*, *"what is the idea of materialization"*, or *"what does 8x certified mean"*, pointing to a severe recall problem of the question-answer database.

Evaluation was performed by manual labeling of top 20 answers retrieved for each of 60 queries for each system by two independent judges. For the sake of consistency, we chose not to use the assessments provided by Jijkoun and de Rijke. Instead, the judges were asked to find agreement on the examples on which they disagreed after each evaluation round. The ratings together with the question-answer pair id were stored and merged into the retrieval results for the next system evaluation. In this way consistency across system evaluations could be ensured, and the effort of manual labeling could be substantially reduced. The quality of retrieval results was assessed according to Jijkoun and de Rijke's (2005) three point scale:

- adequate (2): answer is contained
- material (1): no exact answer, but important information given
- unsatisfactory (0): user's information need is not addressed

The evaluation measure used in Jijkoun and de Rijke (2005) is the success rate at 10 or 20 answers, i.e., $S_2@n$ is the percentage of queries with at least one adequate answer in the top n retrieved question-answer pairs, and $S_{1,2}@n$ is the percentage of queries with at least one adequate or material answer in the top n results. This evaluation measure accounts for improvements in coverage, i.e., it rewards cases where answers are found for queries that did not have an adequate or material answer before. In contrast, the mean reciprocal rank (MRR) measure standardly used in QA can have the effect of preferring systems that find answers only for a small set of queries, but rank them higher than systems with

[4] http://www.metacrawler.com

(1)	query:	how to live with cat allergies
	local expansion (-):	allergens allergic infections filter plasmacluster rhinitis introduction effective replacement
	qa-translation (+):	allergy cats pet food
	paraphrasing (+):	way allergens life allergy feline ways living allergen
(2)	query:	how to design model rockets
	local expansion (-):	models represented orientation drawings analysis element environment different structure
	qa-translation (+):	models rocket
	paraphrasing (+):	missiles missile rocket grenades arrow designing prototype models ways paradigm
(3)	query:	what is dna hybridization
	local expansion (-):	instructions individual blueprint characteristics chromosomes deoxyribonucleic information biological genetic molecule
	qa-translation (+):	slides clone cdna sitting sequences
	paraphrasing (+):	hibridization hybrids hybridation anything hibridacion hybridising adn hybridisation nothing
(4)	query:	how to enhance competitiveness of indian industries
	local expansion (+):	resources production quality processing established investment development facilities institutional
	qa-translation (+):	increase industry
	paraphrasing (+):	promote raise improve increase industry strengthen
(5)	query:	how to induce labour
	local expansion (-):	experience induction practice imagination concentration information consciousness different meditation relaxation
	qa-translation (-):	birth industrial induced induces
	paraphrasing (-):	way workers inducing employment ways labor working child work job action unions

Table 4: Examples for queries and expansion terms yielding improved (+), decreased (-), or unchanged (0) retrieval performance compared to retrieval without expansion.

higher coverage. This makes MRR less adequate for the low-recall setup of FAQ retrieval.

Table 3 shows success rates at 10 and 20 retrieved question-answer pairs for five different systems. The results for the *baseline tfidf* system, following Jijkoun and de Rijke (2005), are shown in row 2. Row 3 presents results for our variant of *local expansion* by pseudo-relevance feedback (Xu and Croft, 1996). Results for *SMT-based expansion* are given in row 4. A comparison of success rates for retrieving at least one adequate answer in the top 10 results shows relative improvements over the baseline of 11.1% for local query expansion, and of 40.7% for combined SMT-based expansion. Success rates at top 20 results show similar relative improvements of 14.2% for local query expansion, and of 22.8% for combined SMT-based expansion. On the easier task of retrieving a material or adequate answer, success rates drop by a small amount for local expansion, and stay unchanged for SMT-based expansion.

These results can be explained by inspecting a few sample query expansions. Examples (1)-(3) in Table 4 illustrate cases where SMT-based query expansion improves results over baseline performance, but local expansion decreases performance by introducing irrelevant terms. In (4) retrieval performance is improved over the baseline for both expansion techniques. In (5) both local and SMT-based expansion introduce terms that decrease retrieval performance compared to retrieval without expansion.

6 Conclusion

We presented two techniques for query expansion in answer retrieval that are based on SMT technology. Our method for question-answer translation uses a large corpus of question-answer pairs extracted from FAQ pages to learn a translation model from questions to answers. SMT-based paraphrasing utilizes large amounts of bilingual data as a new information source to extract phrase-level synonyms. Both SMT-based techniques take the entire query context into account when adding new terms to the original query. In an experimental comparison with a baseline *tfidf* approach and a local query expansion technique on the task of answer retrieval from FAQ pages, we showed a significant improvement of both SMT-based query expansion over both baselines.

Despite the small-scale nature of our current experimental results, we hope to apply the presented techniques to general web retrieval in future work. Another task for future work is to scale up the extraction of question-answer pair data in order to provide an improved resource for question-answer translation.

References

Eugene Agichtein, Steve Lawrence, and Luis Gravano. 2004. Learning to find answers to questions on the web. *ACM Transactions on Internet Technology*, 4(2):129–162.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of (ACL'05)*, Ann Arbor, MI.

Adam L. Berger, Rich Caruana, David Cohn, Dayne Freitag, and Vibhu Mittal. 2000. Bridging the lexical chasm: Statistical approaches to answer-finding. In *Proceedings of SIGIR'00*, Athens, Greece.

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

Robin B. Burke, Kristian J. Hammond, and Vladimir A. Kulyukin. 1997. Question answering from frequently-asked question files: Experiences with the FAQ finder system. *AI Magazine*, 18(2):57–66.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yo-ram Singer. 2006. Online passive-agressive algorithms. *Machine Learning*, 7:551–585.

Pablo Ariel Duboue and Jennifer Chu-Carroll. 2006. Answering the question you wish they had asked: The impact of paraphrasing for question answering. In *Proceedings of (HLT-NAACL'06)*, New York, NY.

Abdessamad Echihabi and Daniel Marcu. 2003. A noisy-channel approach to question answering. In *Proceedings of (ACL'03)*, Sapporo, Japan.

Sanda Harabagiu and Finley Lacatusu. 2004. Strategies for advanced question answering. In *Proceedings of the HLT-NAACL'04 Workshop on Pragmatics of Question Answering*, Boston, MA.

Sanda Harabagiu, Dan Moldovan, Marius Paşca, Rada Mihalcea, Mihai Surdeanu, Răzvan Bunescu, Roxana Gîrju, Vasile Rus, and Paul Morărescu. 2001. The role of lexico-semantic feedback in open-domain textual question-answering. In *Proceedings of (ACL'01)*, Toulouse, France.

Ulf Hermjakob, Abdessamad Echihabi, and Daniel Marcu. 2002. Natural language based reformulation resource and web exploitation for question answering. In *Proceedings of TREC-11*, Gaithersburg, MD.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, and Chin-Yew Lin. 2000. Question answering in webclopedia. In *Proceedings of TREC 9*, Gaithersburg, MD.

Abraham Ittycheriah, Martin Franz, and Salim Roukos. 2001. IBM's statistical question answering system. In *Proceedings of TREC 10*, Gaithersburg, MD.

Valentin Jijkoun and Maarten de Rijke. 2005. Retrieving answers from frequently asked questions pages on the web. In *Proceedings of the Tenth ACM Conference on Information and Knowledge Management (CIKM'05)*, Bremen, Germany.

Thorsten Joachims. 2003. Transductive learning via spectral graph partitioning. In *Proceedings of ICML'03*, Washington, DC.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of (HLT-NAACL'03)*, Edmonton, Cananda.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Journal of Natural Language Engineering*, 7(3):343–360.

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 *Proceedings of (HLT-NAACL'03)*, Edmonton, Cananda.

John Prager, Jennifer Chu-Carroll, and Krysztof Czuba. 2001. Use of wordnet hypernyms for answering what-is questions. In *Proceedings of TREC 10*, Gaithersburg, MD.

Yonggang Qiu and H. P. Frei. 1993. Concept based query expansion. In *Proceedings of SIGIR'93*, Pittsburgh, PA.

Dragomir R. Radev, Hong Qi, Zhiping Zheng, Sasha Blair-Goldensohn, Zhu Zhang, Weigo Fan, and John Prager. 2001. Mining the web for answers to natural language questions. In *Proceedings of (CIKM'01)*, Atlanta, GA.

Radu Soricut and Eric Brill. 2006. Automatic question answering using the web: Beyond the factoid. *Journal of Information Retrieval - Special Issue on Web Information Retrieval*, 9:191–206.

Ellen M. Voorhees. 1994. Query expansion using lexical-semantic relations. In *Proceedings of SIGIR'94*, Dublin, Ireland.

Jinxi Xu and W. Bruce Croft. 1996. Query expansion using local and global document analysis. In *Proceedings of SIGIR'96*, Zurich, Switzerland.

A Computational Model of Text Reuse in Ancient Literary Texts

John Lee
Spoken Language Systems
MIT Computer Science and Artificial Intelligence Laboratory
Cambridge, MA 02139, USA
jsylee@csail.mit.edu

Abstract

We propose a computational model of text reuse tailored for ancient literary texts, available to us often only in small and noisy samples. The model takes into account source alternation patterns, so as to be able to align even sentences with low surface similarity. We demonstrate its ability to characterize text reuse in the Greek New Testament.

1 Introduction

Text reuse is the transformation of a source text into a target text in order to serve a different purpose. Past research has addressed a variety of text-reuse applications, including: journalists turning a news agency text into a newspaper story (Clough et al., 2002); editors adapting an encyclopedia entry to an abridged version (Barzilay and Elhadad, 2003); and plagiarizers disguising their sources by removing surface similarities (Uzuner et al., 2005).

A common assumption in the recovery of text reuse is the conservation of some degree of lexical similarity from the *source sentence* to the *derived sentence*. A simple approach, then, is to define a lexical similarity measure and estimate a score threshold; given a sentence in the target text, if the highest-scoring sentence in the source text is above the threshold, then the former is considered to be derived from the latter. Obviously, the effectiveness of this basic approach depends on the degree of lexical similarity: source sentences that are quoted verbatim are easier to identify than those that have been transformed by a skillful plagiarizer.

The crux of the question, therefore, is how to identify source sentences despite their lack of surface similarity to the derived sentences. Ancient literary texts, which are the focus of this paper, present some distinctive challenges in this respect.

1.1 Ancient Literary Texts

"Borrowed material embedded in the flow of a writer's text is a common phenomenon in Antiquity." (van den Hoek, 1996). Ancient writers rarely acknowledged their sources. Due to the scarcity of books, they often needed to quote from memory, resulting in inexact quotations. Furthermore, they combined multiple sources, sometimes inserting new material or substantially paraphrasing their sources to suit their purpose. To compound the noise, the version of the source text available to us today might not be the same as the one originally consulted by the author. Before the age of the printing press, documents were susceptible to corruptions introduced by copyists.

Identifying the sources of ancient texts is useful in many ways. It helps establish their relative dates. It traces the evolution of ideas. The material quoted, left out or altered in a composition provides much insight into the agenda of its author. Among the more frequently quoted ancient books are the gospels in the New Testament. Three of them — the gospels of Matthew, Mark, and Luke — are called the Synoptic Gospels because of the substantial text reuse among them.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 472–479,
Prague, Czech Republic, June 2007.

Target verses (English translation) Luke 9:30-33	Target verses (original Greek) Luke 9:30-33	Source verses (original Greek) Mark 9:4-5
(9:30) **And**, *behold,* *there* **talked** *with him two men,* *which* **were Moses** *and* **Elias**.	(9:30) **kai** idou andres duo **sunelaloun** autō hoitines **ēsan Mōusēs** kai **Ēlias**	(9:4) kai ōphthē autois **Ēlias** sun **Mōusei** kai **ēsan sullalountes** tō Iēsou
(9:31) *Who appeared in glory, ...* (9:32) *But Peter and they that were with him ...*	(9:31) hoi ophthentes en doxē ... (9:32) ho de Petros kai hoi sun autō ...	(no obvious source verse) (no obvious source verse)
(9:33) **And** *it came to pass,* *as they departed from him,* **Peter** *said unto* **Jesus**, *Master,* **it is good for us to be here:** **and let us make** **three tabernacles; one for thee,** **and one for Moses, and one for Elias:** *not knowing what he said.*	(9:33) **kai** egeneto en tō diachōrizesthai autous ap' autou eipen **ho Petros** pros ton **Iēsoun** epistata **kalon estin hēmas hōde einai** **kai poiēsōmen skēnas treis** **mian soi kai mian Mōusei** **kai mian Ēlia** mē eidōs ho legei	(9:5) **kai** apokritheis **ho Petros** legei tō **Iēsou** rabbi **kalon estin hēmas hōde einai** **kai poiēsōmen treis skēnas** **soi mian kai Mōusei mian** **kai Ēlia mian**

Table 1: Luke 9:30-33 and their source verses in the Gospel of Mark. The Greek words with common stems in the target and source verses are bolded. The King James Version English translation is included for reference. §1.2 comments on the text reuse in these verses.

1.2 Synoptic Gospels

The nature of text reuse among the Synoptics spans a wide spectrum. On the one hand, some revered verses, such as the sayings of Jesus or the apostles, were preserved verbatim. Such is the case with Peter's short speech in the second half of Luke 9:33 (see Table 1). On the other hand, unimportant details may be deleted, and new information weaved in from other sources or oral traditions. For example, "Luke often edits the introductions to new sections with the greatest independence" (Taylor, 1972). To complicate matters, it is believed by some researchers that the version of the Gospel of Mark used by Luke was a more primitive version, customarily called *Proto-Mark*, which is no longer extant (Boismard, 1972). Continuing our example in Table 1, verses 9:31-32 have no obvious counterparts in the Gospel of Mark. Some researchers have attributed them to an earlier version of Mark (Boismard, 1972) or to Luke's "redactional tendencies" (Bovon, 2002).

The result is that some verses bear little resemblance to their sources, due to extensive redaction, or to discrepancies between different versions of the source text. In the first case, any surface similarity score alone is unlikely to be effective. In the second, even deep semantic analysis might not suffice.

1.3 Goals

One property of text reuse that has not been explored in past research is *source alternation patterns*. For example, "it is well known that sections of Luke derived from Mark and those of other origins are arranged in continuous blocks" (Cadbury, 1920). This notion can be formalized with features on the blocks and order of the source sentences. The first goal of this paper is to *leverage source alternation patterns to optimize the global text reuse hypothesis.*

Scholars of ancient texts tend to express their analyses qualitatively. We attempt to translate their insights into a quantitative model. To our best knowledge, this is the first sentence-level, quantitative text-reuse model proposed for ancient texts. Our second goal is thus to *bring a quantitative approach to source analysis of ancient texts.*

2 Previous Work

Text reuse is analyzed at the document level in (Clough et al., 2002), which classifies newspaper articles as wholly, partially, or non-derived from a news agency text. The hapax legomena, and sentence alignment based on N-gram overlap, are found to be the most useful features. Considering a document as a whole mitigates the problem of low similarity scores for some of the derived sentences.

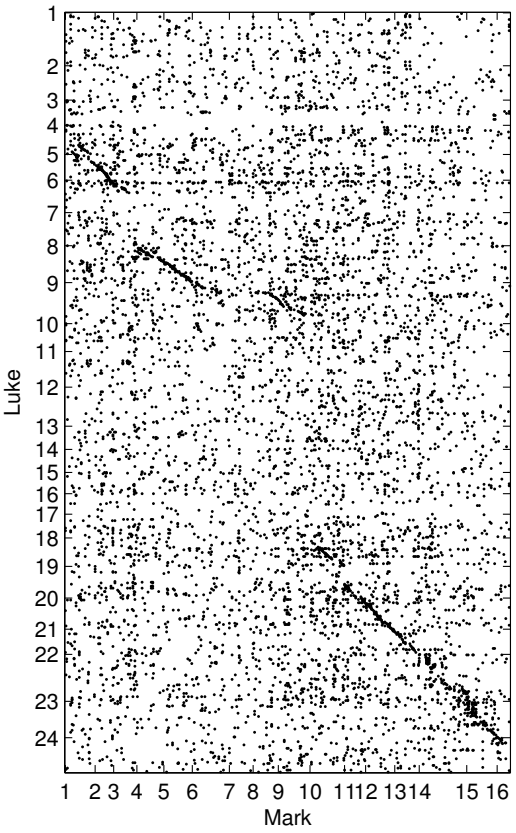


Figure 1: A dot-plot of the cosine similarity measure between the Gospel of Luke and the Gospel of Mark. The number on the axes represent chapters. The thick diagonal lines reflect regions of high lexical similarity between the two gospels.

At the level of short passages or sentences, (Hatzivassiloglou et al., 1999) goes beyond N-gram, taking advantage of WordNet synonyms, as well as ordering and distance between shared words. (Barzilay and Elhadad, 2003) shows that the simple cosine similarity score can be effective when used in conjunction with paragraph clustering. A more detailed comparison with this work follows in §4.2.

In the humanities, reused material in the writings of Plutarch (Helmbold and O'Neil, 1959) and Clement (van den Hoek, 1996) have been manually classified as quotations, reminiscences, references or paraphrases. Studies on the Synoptics have been limited to N-gram overlap, notably (Honoré, 1968) and (Miyake et al., 2004).

Text	Hypothesis	Researcher	Model
L_{train}	$L_{train.B}$	(Bovon, 2002)	*B*
	$L_{train.J}$	(Jeremias, 1966)	*J*
L_{test}	$L_{test.B}$	(Bovon, 2003)	
	$L_{test.J}$	(Jeremias, 1966)	

Table 2: Two models of text reuse of Mark in L_{train} are trained on two different text-reuse hypotheses: The *B* model is on the hypothesis in (Bovon, 2002), and the *J* model, on (Jeremias, 1966). These two models then predict the text-reuse in L_{test}.

3 Data

We assume the Two-Document Theory[1], which hypothesizes that the Gospel of Luke and the Gospel of Matthew have as their common sources two documents: the Gospel of Mark, and a lost text customarily denoted *Q*. In particular, we will consider the Gospel of Luke[2] as the target text, and the Gospel of Mark as the source text.

We use a Greek New Testament corpus prepared by the Center for Computer Analysis of Texts at the University of Pennsylvania[3], based on the text variant from the United Bible Society. The text-reuse hypotheses (i.e., lists of verses deemed to be derived from Mark) of François Bovon (Bovon, 2002; Bovon, 2003) and Joachim Jeremias (Jeremias, 1966) are used. Table 2 presents our notations.

Luke 1:1 to 9:50 (L_{train}, 458 verses) Chapters 1 and 2, narratives of the births of Jesus and John the Baptist, are based on non-Markan sources. Verses 3:1 to 9:50 describe Jesus' activities in Galilee, a substantial part of which is derived from Mark.

Luke Chapters 22 to 24 (L_{test}, 179 verses) These chapters, known as the Passion Narrative, serve as our test text. Markan sources were behind 38% of the verses, according to Bovon, and 7% according to Jeremias.

[1]This theory (Streeter, 1930) is currently accepted by a majority of researchers. It guides our choice of experimental data, but our model does not depend on its validity.

[2]We do not consider the Gospel of Matthew or *Q* in this study. Verses from Luke 9:51 to the end of chapter 21 are also not considered, since their sources are difficult to ascertain (Bovon, 2002).

[3]Obtained through Peter Ballard (personal communication)

4 Approach

For each verse in the target text (a "target verse"), we would like to determine whether it is derived from a verse in the source text (a "source verse") and, if so, which one.

Following the framework of global linear models in (Collins, 2002), we cast this task as learning a mapping F from input verses $\mathbf{x} \in \mathcal{X}$ to a text-reuse hypothesis $\mathbf{y} \in \mathcal{Y} \cup \{\epsilon\}$. $\mathcal{X}$ is the set of verses in the target text. In our case, $\mathbf{x}_{train} = (x_1, \ldots, x_{458})$ is the sequence of verses in L_{train}, and $\mathbf{x}_{test}$ is that of L_{test}. $\mathcal{Y}$ is the set of verses in the source text. Say the sequence $\mathbf{y} = (y_1, \ldots, y_n)$ is the text-reuse hypothesis for $\mathbf{x} = (x_1, \ldots, x_n)$. If y_i is ϵ, then x_i is not derived from the source text; otherwise, y_i is the source verse for x_i. The set of candidates $\mathbf{GEN}(\mathbf{x})$ contains all possible sequences for $\mathbf{y}$, and Θ is the parameter vector. The mapping F is thus:

$$F(\mathbf{x}) = \arg \max_{\mathbf{y} \in \mathbf{GEN}(\mathbf{x})} \Phi(\mathbf{x}, \mathbf{y}) \cdot \Theta$$

4.1 Features

Given the small amount of training data available[4], the feature space must be kept small to avoid overfitting. Starting with the cosine similarity score as the baseline feature, we progressively enrich the model with the following features:

Cosine Similarity [Sim] Treating a target verse as a query to the set of source verses, we compute the cosine similarity, weighted with tf.idf, for each pair of source verse and target verse[5]. This standard bag-of-words approach is appropriate for Greek, a relatively free word-order language. Figure 1 plots this feature on Luke and Mark.

Non-derived verses are assigned a constant score in lieu of the cosine similarity. We will refer to this constant as the *cosine threshold* (C): when the Sim feature alone is used, the constant effectively acts as the threshold above which target verses are considered to be derived. If w_i, w_j are the vectors of words of a target verse and a candidate source verse, then:

$$sim(i,j) = \begin{cases} \frac{w_i \cdot w_j}{\|w_i\| \cdot \|w_j\|} & \text{if derived} \\ C & \text{otherwise} \end{cases}$$

Number of Blocks [Block] Luke can be viewed as alternating between Mark and non-Markan material, and he "prefers to pick up alternatively entire blocks rather than isolated units." (Bovon, 2002) We will use the term *Markan block* to refer to a sequence of verses that are derived from Mark. A verse with a low cosine score, but positioned in the middle of a Markan block, is likely to be derived. Conversely, an isolated verse in the middle of a non-Markan block, even with a high cosine score, is unlikely to be so. The heavier the weight of this feature, the fewer blocks are preferred.

Source Proximity [Prox] When two derived verses are close to one another, their respective source verses are also likely to be close to one another; in other words, derived verses tend to form "continuous blocks" (Cadbury, 1920).

We define *distance* as the number of verses separating two verses. For each pair of consecutive target verses, we take the inverse of the distance between their source verses. This feature is thus intended to discourage a derived verse from being aligned with a source verse that shares some lexical similarities by chance, but is far away from other source verses in the Markan block.

Source Order [Order] "Whenever Luke follows the Markan narrative in his own gospel he follows painstakingly the Markan order", and hence "deviations in the order of the material must therefore be regarded as indications that Luke is not following Mark." (Jeremias, 1966). This feature is a binary function on two consecutive derived verses, indicating whether their source verses are in order. A positive weight for this feature would favor an alignment that respects the order of the source text.

In cases where there are no obvious source verses, such as Luke 9:30-31 in Table 1, the source order

[4]Note that the training set consists of only one $\mathbf{x}_{train}$ — the Gospel of Luke. Luke's only other book, the *Acts of the Apostles*, contains few identifiable reused material.

[5]A targert verse is also allowed to match two consecutive source verses.

and proximity would be disrupted. To mitigate this issue, we allow the `Prox` and `Order` features the option of skipping up to two verses within a Markan block in the target text. In our example, Luke 9:30 can skip to 9:32, preserving the source proximity and order between their source verses, Mark 9:4 and 9:5.

Another potential feature is the occurrence of function words characteristic of Luke (Rehkopf, 1959), along the same lines as in the study of the Federalist Papers (Mosteller and Wallace, 1964). These stylistic indicators, however, are unlikely to be as helpful on the sentence level as on the document level. Furthermore, Luke "reworks [his sources] to an extent that, within his entire composition, the sources rarely come to light in their original independent form" (Bovon, 2002). The significance of the presence of these indicators, therefore, is diminished.

4.2 Discussion

This model is both a simplification of and an extension to the one advocated in (Barzilay and Elhadad, 2003). On the one hand, we perform no paragraph clustering or mapping before sentence alignment. Ancient texts are rarely divided into paragraphs, nor are they likely to be large enough for statistical methods on clustering. Instead, we rely on the `Prox` feature to encourage source verses to stay close to each other in the alignment.

On the other hand, our model makes two extensions to the "Micro Alignment" step in (Barzilay and Elhadad, 2003). First, we add the `Block` and `Prox` features to capture source alternation patterns. Second, we place no hard restrictions on the reordering of the source text, opting instead for a soft preference for maintaining the source order through the `Order` feature. In contrast, deviation from the source order is limited to "flips" between two sentences in (Barzilay and Elhadad, 2003), an assumption that is not valid in the Synoptics[6].

4.3 Evaluation Metric

Our model can make two types of errors: *source error*, when it predicts a non-derived target verse to be derived, or vice versa; and *alignment error*, when it correctly predicts a target verse to be derived, but aligns it to the wrong source verse.

Correspondingly, we interpret the output of our model at two levels: as a binary output, i.e., the target verse is either "derived" or "non-derived"; or, as an alignment of the target verse to a source verse. We measure the precision and recall of the target verses at both levels, yielding two F-measures, F_{source} and F_{align}[7].

Literary dependencies in the Synoptics are typically expressed as pairs of *pericopes* (short, coherent passages), for example, "Luke 22:47-53 // Mark 14:43-52". Likewise, for F_{align}, we consider the output correct if the hypothesized source verse lies within the pericope[8].

5 Experiments

This section presents experiments for evaluating our text-reuse model. §5.1 gives some implementation details. §5.2 describes the training process, which uses text-reuse hypotheses of two different researchers ($L_{train.B}$ and $L_{train.J}$) on the same training text. The two resulting models thus represent two different opinions on how Luke re-used Mark; they then produce two hypotheses on the test text ($\hat{L}_{test.B}$ and $\hat{L}_{test.J}$).

Evaluations of these hypotheses follow. In §5.3, we compare them with the hypotheses of the same two researchers on the test text ($L_{test.B}$ and $L_{test.J}$). In §5.3, we compare them with the hypotheses of seven other representative researchers (Neirynck, 1973). Ideally, when the model is trained on a particular researcher's hypothesis on the train text, its hypothesis on the test text should be closest to the one proposed by the same researcher.

5.1 Implementation

Suppose we align the i^{th} target verse to the k^{th} source verse or to ϵ. Using dynamic programming, their score is the cosine similarity score $sim(i, k)$, added to the best alignment state up to the $(i - 1 - skip)^{th}$ target verse, where $skip$ can vary from 0 to 2 (see §4.1). If the j^{th} source verse is the aligned

[6]For example, Luke 6:12-19 transposes Mark 3:7-12 and Mark 3:13-19 (Bovon, 2002).

[7]Note that F_{align} is never higher than F_{source} since it penalizes both source and alignment errors.

[8]A more fine-grained metric is individual verse alignment. This is unfortunately difficult to measure. As discussed in §1.2, many derived verses have no clear source verses.

Model	*B*		*J*	
Train Hyp	$L_{train.B}$		$L_{train.J}$	
Metric	F_{source}	F_{align}	F_{source}	F_{align}
`Sim`	0.760	0.646	0.748	0.635
`+Block`	0.961	0.728	0.977	0.743
`All`	**0.985**	**0.949**	**0.983**	**0.936**

Table 3: Performance on the training text, L_{train}. The features are accumulative; `All` refers to the full feature set.

verse in this state, then $score(i, k)$ is:

$$sim(i,k) + \max_{j,skip}\{score(i-1-skip, j) + w_{prox} \cdot prox(j,k) + w_{order} \cdot order(j,k) - w_{block} \cdot block(j,k)\}$$

If both j and k are aligned (i.e., not ϵ), then:

$$\begin{aligned} prox(j,k) &= \frac{1}{dist(j,k)} \\ order(j,k) &= 1 \text{ if } j \geq k \\ block(j,k) &= 1 \text{ if starting new block} \end{aligned}$$

Otherwise these are set to zero.

5.2 Training Results

The model takes only four parameters: the weights for the `Block`, `Prox` and `Order` features, as well as the cosine threshold (C). They are empirically optimized, accurate to 0.01, on the two training hypotheses listed in Table 2, yielding two models, *B* and *J*.

Table 3 shows the increasing accuracy of both models in describing the text reuse in L_{train} as more features are incorporated. The `Block` feature contributes most in predicting the block boundaries, as seen in the jump of F_{source} from `Sim` to `+Block`. The `Prox` and `Order` features substantially improve the alignment, boosting the F_{align} from `+Block` to `All`.

Both models *B* and *J* fit their respective hypotheses to very high degrees. For *B*, the only significant source error occurs in Luke 8:1-4, which are derived verses with low similarity scores. They are transitional verses at the beginning of a Markan block. For *J*, the pericope Luke 6:12-16 is wrongly predicted as derived.

Most alignment errors are misalignments to a neighboring pericope, typically for verses located near the boundary between two pericopes. Due to their low similarity scores, the model was unable to decide if they belong to the end of the preceding pericope or to the beginning of the following one.

Model	*B*		*J*	
Test Hyp	$L_{test.B}$		$L_{test.J}$	
Metric	F_{source}	F_{align}	F_{source}	F_{align}
`Sim`	0.579	0.382	0.186	0.144
`+Block`	0.671	0.329	0.743	0.400
`All`	**0.779**	**0.565**	**0.839**	**0.839**

Table 5: Performance on the test text, L_{test}.

5.3 Test Results

The two models trained in §5.2, *B* and *J*, are intended to capture the characteristics of text reuse in L_{train} according to two different researchers. When applied on the test text, L_{test}, they produce two hypotheses, $\hat{L}_{test.B}$ and $\hat{L}_{test.J}$. Ideally, they should be similar to the hypotheses offered by the same researchers (namely, $L_{test.B}$ and $L_{test.J}$), and dissimilar to those by other researchers. We analyze the first aspect in §5.3, and the second aspect in §5.3.

Comparison with Bovon and Jeremias

Table 4 shows the output of *B* and *J* on L_{test}. As more features are added, their output increasingly resemble $L_{test.B}$ and $L_{test.J}$, as shown in Table 5.

Both $\hat{L}_{test.B}$ and $\hat{L}_{test.J}$ contain the same number of Markan blocks as the "reference" hypotheses proposed by the respective scholars. In both cases, the pericope Luke 22:24-30 is correctly assigned as non-derived, despite their relatively high cosine scores. This illustrates the effect of the `Block` feature.

As for source errors, both *B* and *J* mistakenly assign Luke 22:15-18 as Markan, attracted by the high similarity score of Luke 22:18 with Mark 14:25. *B*, in addition, attributes another pericope to Mark where Bovon does not. Despite the penalty of lower source proximity, it wrongly aligned Luke 23:37-38 to Mark 15:2, misled by a specific title of Jesus that happens to be present in both.

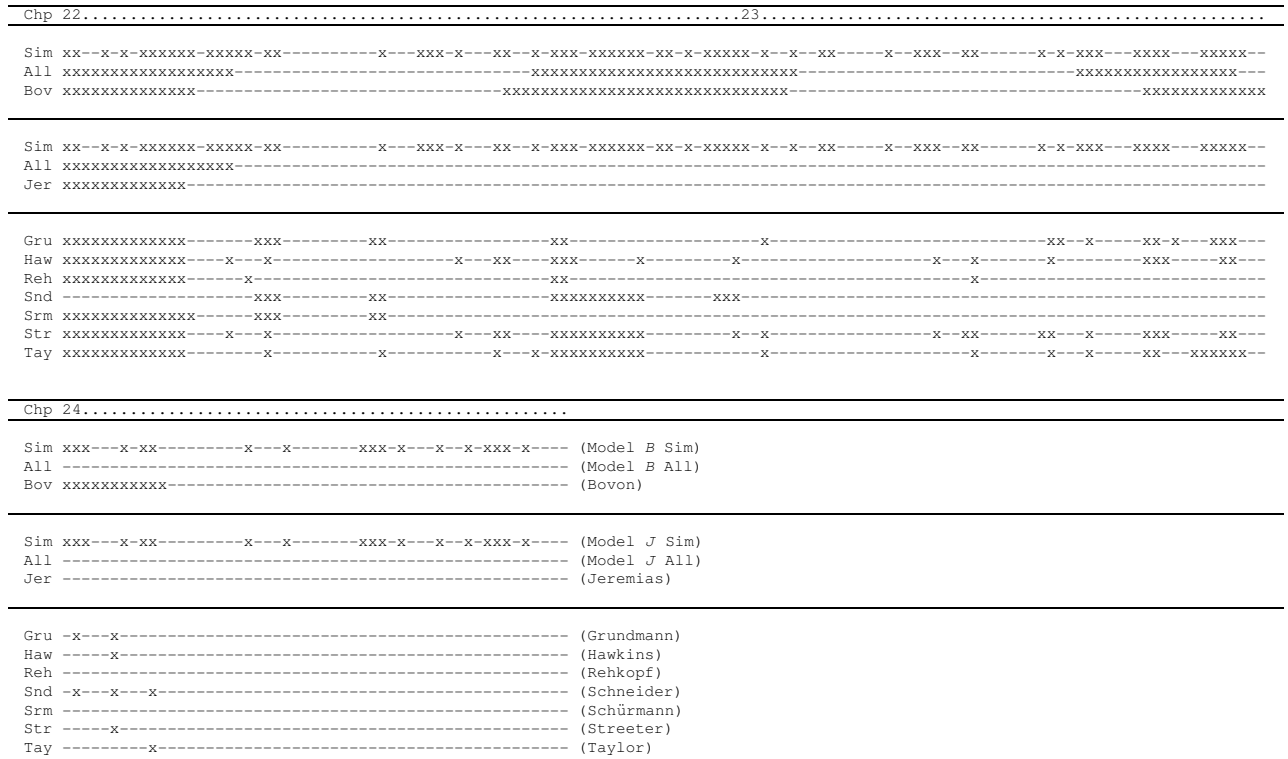


Table 4: Output of models *B* and *J*, and scholarly hypotheses on the test text, L_{test}. The symbol 'x' indicates that the verse is derived from Mark, and '–' indicates that it is not. The hypothesis from (Bovon, 2003), labelled 'Bov', is compared with the Sim (baseline) output and the All output of model *B*, as detailed in Table 5. The hypothesis from (Jeremias, 1966), 'Jer', is similarly compared with outputs of model *J*. Seven other scholarly hypotheses are also listed.

Elsewhere, *B* is more conservative than Bovon in proposing Markan derivation. For instance, the pericope Luke 24:1-11 is deemed non-derived, an opinion (partially) shared by some of the other seven researchers.

Comparison with Other Hypotheses

Another way of evaluating the output of *B* and *J* is to compare them with the hypotheses of other researchers. As shown in Table 6, $\hat{L}_{test.B}$ is more similar to $L_{test.B}$ than to the hypothesis of other researchers[9]. In other words, when the model is trained on Bovon's text-reuse hypothesis on the train text, its prediction on the test text matches most closely with that of the same researcher, Bovon.

Hypothesis	*B* ($\hat{L}_{test.B}$)	*J* ($\hat{L}_{test.J}$)
Bovon ($L_{test.B}$)	**0.838**	0.676
Jeremias ($L_{test.J}$)	0.721	**0.972**
Grundmann	0.726	0.866
Hawkins	0.737	0.877
Rehkopf	0.721	0.950
Schneider	0.676	0.782
Schürmann	0.698	0.950
Streeter	0.771	0.821
Taylor	0.793	0.821

Table 6: Comparison of the output of the models *B* and *J* with hypotheses by prominent researchers listed in (Neirynck, 1973). The metric is the percentage of verses deemed by both hypotheses to be "derived", or "non-derived".

[9]This is the list of researchers whose opinions on L_{test} are considered representative by (Neirynck, 1973). We have simplified their hypotheses, considering those "partially assimilated" and "reflect the influence of Mark" to be non-derived from Mark.

The differences between Bovon and the next two most similar hypotheses, Taylor and Streeter, are not statistically significant according to McNemar's test ($p = 0.27$ and $p = 0.10$ respectively), possibly a reflection of the small size of L_{test}; the differences are significant, however, with all other hypotheses ($p < 0.05$). Similar results are observed for Jeremias and $\hat{L}_{test.J}$.

6 Conclusion & Future Work

We have proposed a text-reuse model for ancient literary texts, with novel features that account for source alternation patterns. These features were validated on the Lukan Passion Narrative, an instance of text reuse in the Greek New Testament.

The model's predictions on this passage are compared to nine scholarly hypotheses. When tuned on the text-reuse hypothesis of a certain researcher on the train text, it favors the hypothesis of the same person on the test text. This demonstrates the model's ability to capture the researcher's particular understanding of text reuse.

While a computational model alone is unlikely to provide definitive answers, it can serve as a supplement to linguistic and literary-critical approaches to text-reuse analysis, and can be especially helpful when dealing with a large amount of candidate source texts.

Acknowledgements

This work grew out of a term project in the course "Gospel of Luke", taught by Professor François Bovon at Harvard Divinity School. It has also benefited much from discussions with Dr. Steven Lulich.

References

R. Barzilay and N. Elhadad. 2003. *Sentence Alignment for Monolingual Comparable Corpora*. Proc. EMNLP.

M. E. Boismard. 1972. *Synopse des quatre Evangiles en français, Tome II*. Editions du Cerf, Paris, France.

F. Bovon. 2002. *Luke I: A Commentary on the Gospel of Luke 1:1-9:50*. Hermeneia. Fortress Press. Minneapolis, MN.

F. Bovon. 2003. The Lukan Story of the Passion of Jesus (Luke 22-23). *Studies in Early Christianity*. Baker Academic, Grand Rapids, MI.

H. J. Cadbury. 1920. *The Style and Literary Method of Luke*. Harvard Theological Studies, Number VI. George F. Moore and James H. Ropes and Kirsopp Lake (ed). Harvard University Press, Cambridge, MA.

P. Clough, R. Gaizauskas, S. S. L. Piao and Y. Wilks. 2002. *METER: MEasuring TExt Reuse*. Proc. ACL.

M. Collins. 2002. *Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms.* Proc. EMNLP.

V. Hatzivassiloglou, J. L. Klavans and E. Eskin. 1999. *Detecting Text Similarity over Short Passages: Exploring Linguistic Feature Combinations via Machine Learning*. Proc. EMNLP.

W. C. Helmbold and E. N. O'Neil. 1959. *Plutarch's Quotations*. Philological Monographs XIX, American Philological Association.

A. M. Honoré. 1968. *A Statistical Study of the Synoptic Problem*. Novum Testamentum, Vol. 10, p.95-147.

J. Jeremias. 1966. *The Eucharistic Words of Jesus*. Scribner's, New York, NY.

M. Miyake, H. Akama, M. Sato, M. Nakagawa and N. Makoshi. 2004. *Tele-Synopsis for Biblical Research: Development of NLP based Synoptic Software for Text Analysis as a Mediator of Educational Technology and Knowledge Discovery*. Proc. IEEE International Conference on Advanced Learning Technologies (ICALT).

F. Mosteller and D. L. Wallace. 1964. *Inference and Disputed Authorship: The Federalist.* Addison Wesley, Reading, MA.

F. Neirynck. 1973. *La matière marcienne dans l'évangile de Luc*. L'Évangile de Luc, Problèmes littéraires et théologiques. Editions Duculot, Belgium.

F. Rehkopf. 1959. *Die lukanische Sonderquelle*. Wissenschaftliche Untersuchungen zum Neuen Testament, Vol. 5. Tübingen, Germany.

B. H. Streeter. 1930. *The Four Gospels: A Study of Origins*. MacMillan. London, England.

V. Taylor. 1972. *The Passion Narrative of St. Luke: A Critical and Historical Investigation*. Society for New Testament Studies Monograph Series, Vol. 19. Cambridge University Press, Cambridge, England.

O. Uzuner, B. Katz and T. Nahnsen. 2005. *Using Syntactic Information to Identify Plagiarism*. Proc. 2nd Workshop on Building Educational Applications using NLP. Ann Arbor, MI.

A. van den Hoek. 1996. *Techniques of Quotation in Clement of Alexandria — A View of Ancient Literary Working Methods*. Vigiliae Christianae, Vol 50, p.223-243. E. J. Brill, Leiden, The Netherlands.

Finding document topics for improving topic segmentation

Olivier Ferret
CEA LIST, LIC2M
18 route du Panorama, BP6
Fontenay aux Roses, F-92265 France
ferreto@zoe.cea.fr

Abstract

Topic segmentation and identification are often tackled as separate problems whereas they are both part of topic analysis. In this article, we study how topic identification can help to improve a topic segmenter based on word reiteration. We first present an unsupervised method for discovering the topics of a text. Then, we detail how these topics are used by segmentation for finding topical similarities between text segments. Finally, we show through the results of an evaluation done both for French and English the interest of the method we propose.

1 Introduction

In this article, we address the problem of linear topic segmentation, which consists in segmenting documents into topically homogeneous segments that does not overlap each other. This part of the Discourse Analysis field has received a constant interest since the initial work in this domain such as (Hearst, 1994). One criterion for classifying topic segmentation systems is the kind of knowledge they depend on. Most of them only rely on surface features of documents: word reiteration in (Hearst, 1994; Choi, 2000; Utiyama and Isahara, 2001; Galley et al., 2003) or discourse cues in (Passonneau and Litman, 1997; Galley et al., 2003). As such systems do not require external knowledge, they are not sensitive to domains but they are limited by the type of documents they can be applied to: lexical reiteration is reliable only if concepts are not too frequently expressed by several means (synonyms, etc.) and discourse cues are often rare and corpus-specific.

To overcome these difficulties, some systems make use of domain-independent knowledge about lexical cohesion: a lexical network built from a dictionary in (Kozima, 1993); a thesaurus in (Morris and Hirst, 1991); a large set of lexical co-occurrences collected from a corpus in (Choi et al., 2001). To a certain extent, these lexical networks enable topic segmenters to exploit a sort of concept reiteration. However, their lack of any explicit topical structure makes this kind of knowledge difficult to use when lexical ambiguity is high.

The most simple solution to this problem is to exploit knowledge about the topics that may occur in documents. Such topic models are generally built from a large set of example documents as in (Yamron et al., 1998), (Blei and Moreno, 2001) or in one component of (Beeferman et al., 1999). These statistical topic models enable segmenters to improve their precision but they also restrict their scope.

Hybrid systems that combine the approaches we have presented were also developed and illustrated the interest of such a combination: (Jobbins and Evett, 1998) combined word recurrence, co-occurrences and a thesaurus; (Beeferman et al., 1999) relied on both lexical modeling and discourse cues; (Galley et al., 2003) made use of word reiteration through lexical chains and discourse cues.

The work we report in this article takes place in the first category we have presented. It does not rely on any *a priori* knowledge and exploits word usage rather than discourse cues. More precisely, we present a new method for enhancing the results

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 480–487,
Prague, Czech Republic, June 2007.

of segmentation systems based on word reiteration without relying on any external knowledge.

2 Principles

In most of the algorithms in the text segmentation field, documents are represented as sequences of basic discourse units. When they are written texts, these units are generally sentences, which is also the case in our work. Each unit is turned into a vector of words, following the principles of the *Vector Space* model. Then, the similarity between the basic units of a text is evaluated by computing a similarity measure between the vectors that represent them. Such a similarity is considered as representative of the topical closeness of the corresponding units. This principle is also applied to groups of basic units, such as text segments, because of the properties of the *Vector Space* model. Segments are finally delimited by locating the areas where the similarity between units or groups of units is weak.

This quick overview highlights the important role of the evaluation of the similarity between discourse units in the segmentation process. When no external knowledge is used, this similarity is only based on the strict reiteration of words. But it can be enhanced by taking into account semantic relations between words. This was done for instance in (Jobbins and Evett, 1998) by taking semantic relations from Roget's Thesaurus. This resource was also used in (Morris and Hirst, 1991) where the similarity between discourse units was more indirectly evaluated through the lexical chains they share. The same approach was adopted in (Stokes et al., 2002) but with WordNet as the reference semantic resource.

In this article, we propose to improve the detection of topical similarity between text segments but without relying on any external knowledge. For each text to segment, we first identify its topics by performing an unsupervised clustering of its words according to their co-occurrents in the text. Thus, each of its topics is represented by a subset of its vocabulary. When the similarity between two segments is evaluated during segmentation, the words they share are first considered but the presence of words of the same topic is also taken into account. This makes it possible to find similar two segments that refer to the same topic although they do not share a lot of words. It is also a way to exploit long-range relations between words at a local level. More globally, it helps to reduce the false detection of topic shifts.

3 Unsupervised Topic Identification

The approach we propose first requires to discover the topics of texts. For performing such a task without using *a priori* knowledge, we assume that the most representative words of each of the topics of a text occur in similar contexts. Hence, for each word of the text with a minimal frequency, we collect its co-occurrents, we evaluate the pairwise similarity of these selected text words by relying on their co-occurrents and finally, we build topics by applying an unsupervised clustering method to them.

3.1 Building the similarity matrix of text words

The first step for discovering the topics of a text is a linguistic pre-processing of it. This pre-processing splits the text into sentences and represents each of them as the sequence of its lemmatized plain words, that is, nouns (proper and common nouns), verbs and adjectives. After filtering the low frequency words of the text (frequency < 3), the co-occurrents of the remaining words are classically collected by recording the co-occurrences in a fixed-size window (15 plain words) moved over the pre-processed text. As a result, each text word is represented by a vector that contains its co-occurrents and their co-occurrence frequency. The pairwise similarity between all the selected text words is then evaluated for building their similarity matrix. We classically apply the *Cosine* measure between the vectors that represent them for this evaluation.

3.2 From a similarity matrix to text topics

The final step for discovering the topics of a text is the unsupervised clustering of its words from their similarity matrix. We rely for this task on an adaptation of the Shared Nearest Neighbor (SNN) algorithm described in (Ertöz et al., 2001). This algorithm particularly fits our needs as it automatically determines the number of clusters – in our case the number of topics of a text – and does not take into account the elements that are not representative of the clusters it builds. This last point is important for our application as all the plain words of a text are not representative of its topics. The SNN algorithm

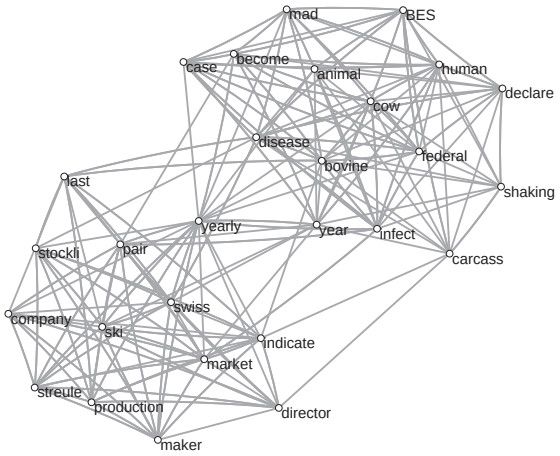


Figure 1: Similarity graph after its sparsification

(see Algorithm 1) performs clustering by detecting high-density areas in a similarity graph. In our case, the similarity graph is directly built from the similarity matrix: each vertex represents a text word and an edge links two words whose similarity is not null. The SNN algorithm splits up into two main stages: the first one finds the elements that are the most representative of their neighborhood. These elements are the seeds of the final clusters that are built in the second stage by aggregating the remaining elements to those selected by the first stage. This first stage

Algorithm 1 SNN algorithm

1. sparsification of the similarity graph
2. building of the SNN graph
3. computation of the distribution of strong links
4. search for topic seeds and filtering of noise
5. building of text topics
6. removal of insignificant topics
7. extension of text topics

starts by sparsifying the similarity graph, which is done by keeping only the links towards the k (k=10) most similar neighbors of each text word (step 1). Figure 1 shows the resulting graph for a two-topic document of our evaluation framework (see Section 5.1). Then, the similarity graph is transposed into a shared nearest neighbor (SNN) graph (step 2). In this graph, the similarity between two words is given by the number of direct neighbors they share in the similarity graph. This transposition makes the similarity values more reliable, especially for high-dimensional data like textual data. Strong links in the SNN graph are finally detected by applying a fixed threshold to the distribution of shared neighbor numbers (step 3). A word with a high number of strong links is taken as the seed of a topic as it is representative of the set of words that are linked to it. On the contrary, a word with few strong links is supposed to be outlier (step 4).

The second stage of the SNN algorithm first builds text topics by associating to topic seeds the remaining words that are the most similar to them provided that their number of shared neighbors is high enough (step 5). Moreover, the seeds that are judged as too close to each other are also grouped during this step in accordance with the same criteria. The last two steps bring small improvements to the results of this clustering. First, when the number of words of a topic is too small (size < 3), this topic is judged as insignificant and it is discarded (step 6). Its words are added to the set of words without topic after step 5. We added this step to the SNN algorithm to balance the fact that without any external knowledge, all the semantic relations between text words cannot be found by relying only on co-occurrence. Finally, the remaining text topics are extended by associating to them the words that are neither noise nor already part of a topic (step 7). As topics are defined at this point more precisely than at step 4, the integration of words that are not strongly linked to a topic seed can be safely performed by relying on the average strength of their links in the SNN graph with the words of the topic. After the SNN algorithm is applied, a set of topics is associated to the text to segment, each of them being defined as a subset of its vocabulary.

4 Using Text Topics for Segmentation

4.1 Topic segmentation using word reiteration

As *TextTiling*, the topic segmentation method of Hearst (Hearst, 1994), the topic segmenter we propose, called F06, first evaluates the lexical cohesion of texts and then finds their topic shifts by identifying breaks in this cohesion. The first step of this process is the linguistic pre-processing of texts, which is identical for topic segmentation to the pre-

processing described in Section 3.1 for the discovering of text topics. The evaluation of the lexical cohesion of a text relies as for *TextTiling* on a fixed-size focus window that is moved over the text to segment and stops at each sentence break. The cohesion in the part of text delimited by this window is evaluated by measuring the word reiteration between its two sides. This is done in our case by applying the *Dice coefficient* between the two sides of the focus window, following (Jobbins and Evett, 1998). This cohesion value is associated to the sentence break at the transition between the two sides of the window. More precisely, if W_l refers to the vocabulary of the left side of the focus window and W_r refers to the vocabulary of its right side, the cohesion in the window at position x is given by:

$$LC_{rec}(x) = \frac{2 \cdot card(W_l \cap W_r)}{card(W_l) + card(W_r)} \quad (1)$$

This measure was adopted instead of the *Cosine* measure used in *TextTiling* because its definition in terms of sets makes it easier to extend for taking into account other types of relations, as in (Jobbins and Evett, 1998). A cohesion value is computed for each sentence break of the text to segment and the final result is a cohesion graph of the text.

The last part of our algorithm is mainly taken from the *LCseg* system (Galley et al., 2003) and is divided into three steps:

- computation of a score evaluating the probability of each minimum of the cohesion graph to be a topic shift;
- removal of segments with a too small size;
- selection of topic shifts.

The computation of the score of a minimum m begins by finding the pair of maxima l and r around it. This score is then given by:

$$score(m) = \frac{LC(l) + LC(r) - 2 \cdot LC(m)}{2} \quad (2)$$

This score, whose values are between 0 and 1, is a measure of how high is the difference between the minimum and the maxima around it. Hence, it favors as possible topic shifts minima that correspond to sharp falls of lexical cohesion.

The next step is done by removing as a possible topic shift each minimum that is not farther than 2 sentences from its preceding neighbor. Finally, the selection of topic shifts is performed by applying a threshold computed from the distribution of minimum scores. Thus, a minimum m is kept as a topic shift if $score(m) > \mu - \alpha \cdot \sigma$, where μ is the average of minimum scores, σ their standard deviation and α is a modulator ($\alpha = 0.6$ in our experiments).

4.2 Using text topics to enhance segmentation

The heart of the algorithm we have presented above is the evaluation of lexical cohesion in the focus window, as given by Equation 1. This evaluation is also a weak point as $card(W_l \cap W_r)$ only relies on word reiteration. As a consequence, two different words that respectively belongs to W_l and W_r but also belong to the same text topic cannot contribute to the identification of a possible topical similarity between the two sides of the focus window.

The algorithm F06T is based on the same principles as F06 but it extends the evaluation of lexical cohesion by taking into account the topical proximity of words. The reference topics for judging this proximity are of course the text topics discovered by the method of Section 3. In this extended version, the evaluation of the cohesion in the focus window is made of three steps:

- computation of the word reiteration cohesion;
- determination of the topic(s) of the window;
- computation of the cohesion based on text topics and fusion of the two kinds of cohesion.

The first step is identical to the computation of the cohesion in F06. The second one aims at restricting the set of topics that are used in the last step to the topics that are actually representative of the content of the focus window, *i.e.* representative of the current context of discourse. This point is especially important in the areas where the current topic is changing because amplifying the influence of the surrounding topics can lead to the topic shift being missed. Hence, a topic is considered as representative of the content of the focus window only if it matches each side of this window. In practice, this matching is evaluated by applying the *Cosine* measure between the vector that represents one side of

the window and the vector that represents the topic[1] and by testing if the resulting value is higher than a fixed threshold (equal to 0.1 in the experiments of Section 5). It must be noted that several topics may be associated to the focus window. As the discovering of text topics is done in an unsupervised way and without any external knowledge, a theme of a text may be scattered over several identified topics and then, its presence can be characterized by several of them.

The last step of the cohesion evaluation first consists in determining for each side of the focus window the number of its words that belong to one of the topics associated to the window. The cohesion of the window is then given by Equation 3, that estimates the significance of the presence of the text topics in the window:

$$LC_{top}(x) = \frac{card(TW_l) + card(TW_r)}{card(W_l) + card(W_r)} \qquad (3)$$

where $TW_{i \in \{l,r\}} = (W_i \cap T_w) - (W_l \cap W_r)$ and T_w is the union of all the representations of the topics associated to the window. TW_i corresponds to the words of the i side of the window that belong to the topics of the window ($W_i \cap T_w$) but are not part of the vocabulary from which the lexical cohesion based on word reiteration is computed ($W_l \cap W_r$).

Finally, the global cohesion in the focus window is computed as the sum of the two kinds of cohesion, the one computed from word reiteration (see Equation 1) and the one computed from text topics (see Equation 3).

5 Evaluation

5.1 Evaluation framework

The main objective of our evaluation was to verify that taking into account text topics discovered without relying on external knowledge can actually improve a topic segmentation algorithm that is initially based on word reiteration. Since the work of Choi (Choi, 2000), the evaluation framework he proposed has become a kind of standard for the evaluation of topic segmentation algorithms. This framework is based on the building of artificial texts made of segments extracted from different documents. It has at least two advantages: the reference corpus is easy to build as it does not require human annotations; parameters such as the size of the documents or the segments can be precisely controlled. But it has also an obvious drawback: its texts are artificial. This is a problem in our case as our algorithm for discovering text topics exploits the fact that the words of a topic tend to co-occur at the document scale. This hypothesis is no longer valid for documents built according to the procedure of Choi. It is why we adapted his framework for having more realistic documents without losing its advantages. This adaptation concerns the way the document segments are selected. Instead of taking each segment from a different document, we only use two source documents. Each of them is split into a set of segments whose size is between 3 and 11 sentences, as for Choi, and an evaluation document is built by concatenating these segments in an alternate way from the beginning of the source documents, *i.e.* one segment from a source document and the following from the other one, until 10 segments are extracted. Moreover, in order to be sure that the boundary between two adjacent segments of an evaluation document actually corresponds to a topic shift, the source documents are selected in such a way that they refer to different topics. This point was controlled in our case by taking documents from the corpus of the CLEF 2003 evaluation for crosslingual information retrieval: each evaluation document was built from two source documents that had been judged as relevant for two different CLEF 2003 topics. Two evaluation corpora made of 100 documents each, one in French and one in English, were built following this procedure. Table 1 shows their main characteristics.

	French	**English**
# source doc.	128	87
# source topics	11	3
segments/doc.	10 (84%) 8 (16%)	10 (97%) 8 (3%)
sentences/doc.	65	68
plain words/doc.	797	604

Table 1: Data about our evaluation corpora

[1] Each word of the topic vector has a weight equal to 1. In the window vector, this weight is equal to the frequency of the word in the corresponding side of the window.

5.2 Topic identification

As F06T exploits document topics, we also evaluated our method for topic identification. This evaluation is based on the corpus of the previous section. For each of its documents, a reference topic is built from each group of segments that come from the same source document by gathering the words that only appear in these segments. A reference topic is associated to the discovered topic that shares with it the largest number of words. Three complementary measures were computed to evaluate the quality of discovered topics. The main one is purity, which is classically used for unsupervised clustering:

$$Purity = \sum_{i=1}^{k} \frac{v_i}{V} P(Td_i) \quad (4)$$

where $P(Td_i)$, the purity of the discovered topic Td_i, is equal to the fraction of the vocabulary of Td_i that is part of the vocabulary of the reference topic Td_i is assigned to, V is the vocabulary of all the discovered topics and v_i is the vocabulary of Td_i. The second measure evaluates to what extent the reference topics are represented among the discovered topics and is equal to the ratio between the number of discovered topics that are assigned to a reference topic (*assigned discovered topics*) and the number of reference topics. The last measure estimates how strongly the vocabulary of reference topics is present among the discovered topics and is equal to the ratio between the size of the vocabulary of the assigned discovered topics and the size of the vocabulary of reference topics. Table 2 gives the mean

	purity	**reference topics (%)**	**ref. topic vocab. (%)**
French	0.771 (0.117)	89.5 (23.9)	29.9 (7.8)
English	0.766 (0.082)	99.0 (10.0)	31.6 (5.3)

Table 2: Evaluation of topic identification

of each measure, followed by its standard deviation. Results are globally similar for French and English. They show that our method for topic identification builds topics that are rather pure, *i.e.* each of them is strongly tied to a reference topic, but their content is rather sparse in comparison with the content of their associated reference topics.

5.3 Topic segmentation

For validating the hypothesis that underlies our work, we applied F06 and F06T to find the topic bounds in the documents of our two evaluation corpora. Moreover, we also tested four well known segmenters on our corpora to compare the results of F06 and F06T with state-of-the-art algorithms. We classically used the error metric P_k proposed in (Beeferman et al., 1999) to measure segmentation accuracy. P_k evaluates the probability that a randomly chosen pair of sentences, separated by k sentences, is wrongly classified, *i.e.* they are found in the same segment while they are actually in different ones (miss) or they are found in different segments while they are actually in the same one (false alarm). We also give the value of *WindowDiff* (WD), a variant of P_k proposed in (Pevzner and Hearst, 2002) that corrects some of its insufficiencies. Tables 3 and 4 show

systems	$\mathbf{P_k}$	$\mathbf{p_{val}}$**(F06)**	$\mathbf{p_{val}}$**(F06T)**	**WD**
U00	25.91	**0.003**	**1.3e-07**	27.42
C99	27.57	**4.2e-05**	**3.6e-10**	35.42
TextTiling*	21.08	0.699	**0.037**	27.43
LCseg	20.55	0.439	0.111	28.31
F06	21.58	⁄	**0.013**	27.83
F06T	*18.46*	**0.013**	⁄	*24.05*

Table 3: Evaluation of topic segmentation for the French corpus (P_k and WD as percentages)

the results of our evaluations for topic segmentation (smallest values are best results). U00 is the system described in (Utiyama and Isahara, 2001), C99 the one proposed in (Choi, 2000) and *LCseg* is presented in (Galley et al., 2003). *TextTiling** is a variant of *TextTiling* in which the final identification of topic shifts is taken from (Galley et al., 2003). All these systems were used as F06 and F06T without fixing the number of topic shifts to find. Moreover, their parameters were tuned for our evaluation corpus to obtain their best results. For each result, we also give the significance level p_{val} of its difference for P_k with F06 and F06T, evaluated by a one-side t-test with a null hypothesis of equal means. Levels lower than 0.05 are considered as statistically significant (bold-faced values). The first important point to notice about these tables is the fact that

systems	P_k	p_{val}(F06)	p_{val}(F06T)	WD
U00	19.42	**0.048**	**4.3e-05**	21.22
C99	21.63	**1.2e-04**	**1.8e-09**	30.64
TextTiling*	15.81	0.308	0.111	19.80
LCseg	14.78	**0.043**	0.496	19.73
F06	16.90	/	**0.010**	20.93
F06T	*14.06*	**0.010**	/	*18.31*

Table 4: Evaluation of topic segmentation for the English corpus (P_k and WD as percentages)

F06T has significantly better results than F06, both for French and English. Hence, it confirms our hypothesis about the interest of taking into account the topics of a text for its segmentation, even if these topics were discovered in an unsupervised way and without using external knowledge. Moreover, F06T have the best results among all the tested algorithms, with a significant difference in most of the cases.

Another notable point about these results is their stability across our two corpora, even if these corpora are quite similar. Whereas F06 and F06T were initially developed on a corpus in French, their results on the English corpus are comparable to their results on the French test corpus, both for the difference between them and the difference with the four other algorithms. The comparison with these algorithms also illustrates the relationships between them: *TextTiling**, *LCseg*, F06 and F06T share a large number of principles and their overall results are significantly higher than the results of U00 and C99. This trend is different from the one observed from the Choi corpus for which algorithms such C99 or U00 have good results (P_k for C99, U00, F06 and F06T is respectively equal to 12%, 10%, 14% and 14%). This means probably that algorithms with good results on a corpus built as the Choi corpus will not necessarily have good results on "true" texts, which agrees with (Georgescul et al., 2006). Finally, we can observe that all these algorithms have better results on the English corpus than on the French one. As the two corpora are quite similar, this difference seems to come from their difference of language, perhaps because repetitions are more discouraged in French than in English from a stylistic viewpoint. This tends to be confirmed by the ratio between the size of the lemmatized vocabulary of each corpus and their number of tokens, equal to 8% for the French corpus and to 5.6% for the English corpus.

6 Related Work

One of the main problems addressed by our work is the detection of the topical similarity of two text units. We have tackled this problem following an endogenous approach, which is new in the topic segmentation field to our knowledge. The main advantage of this option is that it does not require external knowledge. Moreover, it can integrate relations between words, such as proper nouns for instance, that are unlikely to be found in an external resource.

Other solutions have been already proposed to solve the problem we consider. Most of them consist of two steps: first, they automatically build a semantic representation of words from the co-occurrences collected from a large corpus; then, they use this representation for enhancing the representation of each text unit to compare. This overall principle is implemented with different forms by several topic segmenters. In CWM (Choi et al., 2001), a variant of C99, each word of a sentence is replaced by its representation in a *Latent Semantic Analysis* (LSA) space. In the work of Ponte and Croft (Ponte and Croft, 1997), the representations of sentences are expanded by adding to them words selected from an external corpus by the means of the *Local Context Analysis* (LCA) method. Finally in (Caillet et al., 2004), a set of concepts are learnt from a corpus in an unsupervised way by using the X-means clustering algorithm and the paragraphs of documents are represented in the space defined by these concepts. In fact, the way we use relations between words is closer to (Jobbins and Evett, 1998), even if the relations in this work come from a network of co-occurrences or a thesaurus rather than from text topics. In both cases the similarity of two text units is determined by the proportion of their words that are part of a relation across the two units.

More globally, our work exploits the topics of a text for its segmentation. This kind of approach was also explored in (Blei and Moreno, 2001) where probabilistic topic models were built in an unsupervised way. More recently, (Purver et al., 2006) has also proposed a method for unsupervised topic modeling to address both topic segmentation and identi-

fication. (Purver et al., 2006) is closer to our work than (Blei and Moreno, 2001) because it does not require to build topic models from a corpus but as in our case, its results do not outperform *LCseg* (Galley et al., 2003) while its model is far more complex.

7 Conclusion and Future Work

In this article, we have first proposed an unsupervised method for discovering the topics of a text without relying on external knowledge. Then, we have shown how these topics can be used for improving a topic segmentation method based on word reiteration. Moreover, we have proposed an adaptation of the evaluation framework of Choi that aims at building more realistic evaluation documents. Finally, we have demonstrated the interest of the method we present through its evaluation both on a French and an English corpus.

However, the solution we have proposed for improving the identification of topical similarities between text excerpts cannot completely make up for not using any external knowledge. Hence, we plan to use a network of lexical co-occurrences, which is a source of knowledge that is easy to build automatically from a large corpus. More precisely, we intend to extend our method for discovering text topics by combining the co-occurrence graph of a document with such a network. This network could also be used more directly for topic segmentation as in (Jobbins and Evett, 1998).

References

Doug Beeferman, Adam Berger, and John Lafferty. 1999. Statistical models for text segmentation. *Machine Learning*, 34(1):177–210.

David M. Blei and Pedro J. Moreno. 2001. Topic segmentation with an aspect hidden markov model. In *24th ACM SIGIR*, pages 343–348.

Marc Caillet, Jean-François Pessiot, Massih Amini, and Patrick Gallinari. 2004. Unsupervised learning with term clustering for thematic segmentation of texts. In *RIAO'04*, pages 1–11.

Freddy Y. Y. Choi, Peter Wiemer-Hastings, and Johanna Moore. 2001. Latent semantic analysis for text segmentation. In *EMNLP'01*, pages 109–117.

Freddy Y. Y. Choi. 2000. Advances in domain independent linear text segmentation. In *NAACL'00*, pages 26–33.

Levent Ertöz, Michael Steinbach, and Vipin Kuma. 2001. Finding topics in collections of documents: A shared nearest neighbor approach. In *Text Mine'01, Workshop of the 1st SIAM International Conference on Data Mining*.

Michel Galley, Kathleen McKeown, Eric Fosler-Lussier, and Hongyan Jing. 2003. Discourse segmentation of multi-party conversation. In *ACL'03*, pages 562–569.

Maria Georgescul, Alexander Clark, and Susan Armstrong. 2006. An analysis of quantitative aspects in the evaluation of thematic segmentation algorithms. In *7th SIGdial Workshop on Discourse and Dialogue*, pages 144–151.

Marti A. Hearst. 1994. Multi-paragraph segmentation of expository text. In *ACL'94*, pages 9–16.

Amanda C. Jobbins and Lindsay J. Evett. 1998. Text segmentation using reiteration and collocation. In *ACL-COLING'98*, pages 614–618.

Hideki Kozima. 1993. Text segmentation based on similarity between words. In *ACL'93 (Student Session)*, pages 286–288.

Jane Morris and Graeme Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17(1):21–48.

Rebecca J. Passonneau and Diane J. Litman. 1997. Discourse segmentation by human and automated means. *Computational Linguistics*, 23(1):103–139.

Lev Pevzner and Marti A. Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36.

Jay M. Ponte and Bruce W. Croft. 1997. Text segmentation by topic. In *First European Conference on research and advanced technology for digital libraries*.

Matthew Purver, Konrad P. Körding, Thomas L. Griffiths, and Joshua B. Tenenbaum. 2006. Unsupervised topic modelling for multi-party spoken discourse. In *COLING-ACL 2006*, pages 17–24.

N. Stokes, J. Carthy, and A.F. Smeaton. 2002. Segmenting broadcast news streams using lexical chains. In *STAIRS'02*, pages 145–154.

Masao Utiyama and Hitoshi Isahara. 2001. A statistical model for domain-independent text segmentation. In *ACL'01*, pages 491–498.

J.P. Yamron, I. Carp, L. Gillick, S. Lowe, and P. van Mulbregt. 1998. A hidden markov model approach to text segmentation and event tracking. In *ICASSP*, pages 333–336.

The utility of parse-derived features for automatic discourse segmentation

Seeger Fisher and Brian Roark
Center for Spoken Language Understanding, OGI School of Science & Engineering
Oregon Health & Science University, Beaverton, Oregon, 97006 USA
{fishers,roark}@cslu.ogi.edu

Abstract

We investigate different feature sets for performing automatic sentence-level discourse segmentation within a general machine learning approach, including features derived from either finite-state or context-free annotations. We achieve the best reported performance on this task, and demonstrate that our SPADE-inspired context-free features are critical to achieving this level of accuracy. This counters recent results suggesting that purely finite-state approaches can perform competitively.

1 Introduction

Discourse structure annotations have been demonstrated to be of high utility for a number of NLP applications, including automatic text summarization (Marcu, 1998; Marcu, 1999; Cristea et al., 2005), sentence compression (Sporleder and Lapata, 2005), natural language generation (Prasad et al., 2005) and question answering (Verberne et al., 2006). These annotations include sentence segmentation into discourse units along with the linking of discourse units, both within and across sentence boundaries, into a labeled hierarchical structure. For example, the tree in Figure 1 shows a sentence-level discourse tree for the string "Prices have dropped but remain quite high, according to CEO Smith," which has three discourse segments, each labeled with either "Nucleus" or "Satellite" depending on how central the segment is to the coherence of the text.

There are a number of corpora annotated with discourse structure, including the well-known RST Treebank (Carlson et al., 2002); the Discourse GraphBank (Wolf and Gibson, 2005); and the Penn Discourse Treebank (Miltsakaki et al., 2004). While the annotation approaches differ across these corpora, the requirement of sentence segmentation into sub-sentential discourse units is shared across all approaches. These resources have facilitated research into stochastic models and algorithms for automatic discourse structure annotation in recent years.

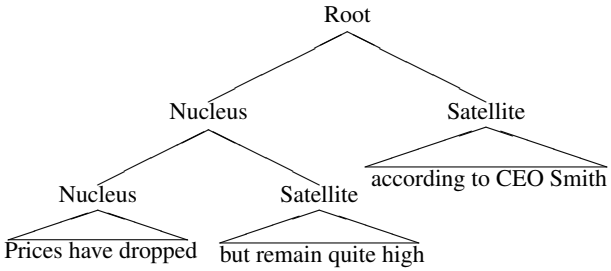


Figure 1: Example Nucleus/Satellite labeled sentence-level discourse tree.

Using the RST Treebank as training and evaluation data, Soricut and Marcu (2003) demonstrated that their automatic sentence-level discourse parsing system could achieve near-human levels of accuracy, if it was provided with manual segmentations and manual parse trees. Manual segmentation was primarily responsible for this performance boost over their fully automatic system, thus making the case that automatic discourse segmentation is the primary impediment to high accuracy automatic sentence-level discourse structure annotation. Their models and algorithm – subsequently packaged together into the publicly available SPADE discourse parser[1] – make use of the output of the Charniak (2000) parser to derive syntactic indicator features for segmentation and discourse parsing.

Sporleder and Lapata (2005) also used the RST Treebank as training data for data-driven discourse parsing algorithms, though their focus, in contrast to Soricut and Marcu (2003), was to avoid context-free parsing and rely exclusively on features in their model that could be derived via finite-state chunkers and taggers. The annotations that they derive are dis-

[1] http://www.isi.edu/publications/licensed-sw/spade/

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 488–495,
Prague, Czech Republic, June 2007.

course "chunks", i.e., sentence-level segmentation and non-hierarchical nucleus/span labeling of segments. They demonstrate that their models achieve comparable results to SPADE without the use of any context-free features. Once again, segmentation is the part of the process where the automatic algorithms most seriously underperform.

In this paper we take up the question posed by the results of Sporleder and Lapata (2005): how much, if any, accuracy reduction should we expect if we choose to use only finite-state derived features, rather than those derived from full context-free parses? If little accuracy is lost, as their results suggest, then it would make sense to avoid relatively expensive context-free parsing, particularly if the amount of text to be processed is large or if there are real-time processing constraints on the system. If, however, the accuracy loss is substantial, one might choose to avoid context-free parsing only in the most time-constrained scenarios.

While Sporleder and Lapata (2005) demonstrated that their finite-state system could perform as well as the SPADE system, which uses context-free parse trees, this does not directly answer the question of the utility of context-free derived features for this task. SPADE makes use of a particular kind of feature from the parse trees, and does not train a general classifier making use of other features beyond the parse-derived indicator features. As we shall show, its performance is not the highest that can be achieved via context-free parser derived features.

In this paper, we train a classifier using a general machine learning approach and a range of finite-state and context-free derived features. We investigate the impact on discourse segmentation performance when one feature set is used versus another, in such a way establishing the utility of features derived from context-free parses. In the course of so doing, we achieve the best reported performance on this task, an absolute F-score improvement of 5.0% over SPADE, which represents a more than 34% relative error rate reduction.

By focusing on segmentation, we provide an approach that is generally applicable to all of the various annotation approaches, given the similarities between the various sentence-level segmentation guidelines. Given that segmentation has been shown to be a primary impediment to high accuracy sentence-level discourse structure annotation, this represents a large step forward in our ability to automatically parse the discourse structure of text, whatever annotation approach we choose.

2 Methods

2.1 Data

For our experiments we use the Rhetorical Structure Theory Discourse Treebank (Carlson et al., 2002), which we will denote RST-DT, a corpus annotated with discourse segmentation and relations according to Rhetorical Structure Theory (Mann and Thompson, 1988). The RST-DT consists of 385 documents from the Wall Street Journal, about 176,000 words, which overlaps with the Penn Wall St. Journal (WSJ) Treebank (Marcus et al., 1993).

The segmentation of sentences in the RST-DT is into clause-like units, known as elementary discourse units, or *edus*. We will use the two terms '*edu*' and 'segment' interchangeably throughout the rest of the paper. Human agreement for this segmentation task is quite high, with agreement between two annotators at an F-score of 98.3 for unlabeled segmentation (Soricut and Marcu, 2003).

The RST-DT corpus annotates *edu* breaks, which typically include sentence boundaries, but sentence boundaries are not explicitly annotated in the corpus. To perform sentence-level processing and evaluation, we aligned the RST-DT documents to the same documents in the Penn WSJ Treebank, and used the sentence boundaries from that corpus.[2] An additional benefit of this alignment is that the Penn WSJ Treebank tokenization is then available for parsing purposes. Simple minimum edit distance alignment effectively allowed for differences in punctuation representation (e.g., double quotes) and tokenization when deriving the optimal alignment.

The RST-DT corpus is partitioned into a training set of 347 documents and a test set of 38 documents. This test set consists of 991 sentences with 2,346 segments. For training purposes, we created a held-out development set by selecting every tenth sentence of the training set. This development set was used for feature development and for selecting the number of iterations used when training models.

2.2 Evaluation

Previous research into RST-DT segmentation and parsing has focused on subsets of the 991 sentence test set during evaluation. Soricut and Marcu (2003)

[2] A small number of document final parentheticals are in the RST-DT and not in the Penn WSJ Treebank, which our alignment approach takes into account.

omitted sentences that were not exactly spanned by a subtree of the treebank, so that they could focus on sentence-level discourse parsing. By our count, this eliminates 40 of the 991 sentences in the test set from consideration. Sporleder and Lapata (2005) went further and established a smaller subset of 608 sentences, which omitted sentences with only one segment, i.e., sentences which themselves are atomic *edus*.

Since the primary focus of this paper is on segmentation, there is no strong reason to omit any sentences from the test set, hence our results will evaluate on all 991 test sentences, with two exceptions. First, in Section 2.3, we compare SPADE results under our configuration with results from Sporleder and Lapata (2005) in order to establish comparability, and this is done on their 608 sentence subset. Second, in Section 3.2, we investigate feeding our segmentation into the SPADE system, in order to evaluate the impact of segmentation improvements on their sentence-level discourse parsing performance. For those trials, the 951 sentence subset from Soricut and Marcu (2003) is used. All other trials use the full 991 sentence test set.

Segmentation evaluation is done with precision, recall and F1-score of segmentation boundaries. Given a word string $w_1 \ldots w_k$, we can index word boundaries from 0 to k, so that each word w_i falls between boundaries $i-1$ and i. For sentence-based segmentation, indices 0 and k, representing the beginning and end of the string, are known to be segment boundaries. Hence Soricut and Marcu (2003) evaluate with respect to sentence internal segmentation boundaries, i.e., with indices j such that $0<j<k$ for a sentence of length k. Let g be the number of sentence-internal segmentation boundaries in the gold standard, t the number of sentence-internal segmentation boundaries in the system output, and m the number of correct sentence-internal segmentation boundaries in the system output. Then

$$P = \frac{m}{t} \quad R = \frac{m}{g} \quad \text{and} \quad F1 = \frac{2PR}{P+R} = \frac{2m}{g+t}$$

In Sporleder and Lapata (2005), they were primarily interested in labeled segmentation, where the segment initial boundary was labeled with the segment type. In such a scenario, the boundary at index 0 is no longer known, hence their evaluation included those boundaries, even when reporting unlabeled results. Thus, in section 2.3, for comparison with reported results in Sporleder and Lapata (2005), our F1-score is defined accordingly, i.e., segmentation boundaries j such that $0 \leq j < k$.

Segmentation system	F1
Sporleder and Lapata best (reported)	88.40
SPADE	
Sporleder and Lapata configuration (reported):	87.06
current configuration:	91.04

Table 1: Segmentation results on the Sporleder and Lapata (2005) data set, with accuracy defined to include sentence initial segmentation boundaries.

In addition, we will report unlabeled bracketing precision, recall and F1-score, as defined in the PARSEVAL metrics (Black et al., 1991) and evaluated via the widely used *evalb* package. We also use *evalb* when reporting labeled and unlabeled discourse parsing results in Section 3.2.

2.3 Baseline SPADE setup

The publicly available SPADE package, which encodes the approach in Soricut and Marcu (2003), is taken as the baseline for this paper. We made several modifications to the script from the default, which account for better baseline performance than is achieved with the default configuration. First, we modified the script to take given parse trees as input, rather than running the Charniak parser itself. This allowed us to make two modifications that improved performance: turning off tokenization in the Charniak parser, and reranking. The default script that comes with SPADE does not turn off tokenization inside of the parser, which leads to degraded performance when the input has already been tokenized in the Penn Treebank style. Secondly, Charniak and Johnson (2005) showed how reranking of the 50-best output of the Charniak (2000) parser gives substantial improvements in parsing accuracy. These two modifications to the Charniak parsing output used by the SPADE system lead to improvements in its performance compared to previously reported results.

Table 1 compares segmentation results of three systems on the Sporleder and Lapata (2005) 608 sentence subset of the evaluation data: (1) their best reported system; (2) the SPADE system results reported in that paper; and (3) the SPADE system results with our current configuration. The evaluation uses the unlabeled F1 measure as defined in that paper, which counts sentence initial boundaries in the scoring, as discussed in the previous section. As can be seen from these results, our improved configuration of SPADE gives us large improvements over the previously reported SPADE performance on this subset. As a result, we feel that we can use SPADE

as a very strong baseline for evaluation on the entire test set.

Additionally, we modified the SPADE script to allow us to provide our segmentations to the full discourse parsing that it performs, in order to evaluate the improvements to discourse parsing yielded by any improvements to segmentation.

2.4 Segmentation classifier

For this paper, we trained a binary classifier, which was applied independently at each word w_i in the string $w_1 \dots w_k$, to decide whether that word is the last in a segment. Note that w_k is the last word in the string, and is hence ignored. We used a log-linear model with no Markov dependency between adjacent tags,[3] and trained the parameters of the model with the perceptron algorithm, with averaging to control for over-training (Collins, 2002).

Let $C=\{E, I\}$ be the set of classes: segmentation boundary (E) or non-boundary (I). Let $f(c, i, w_1 \dots w_k)$ be a function that takes as input a class value c, a word index i and the word string $w_1 \dots w_k$ and returns a d-dimensional vector of feature values for that word index in that string with that class. For example, one feature might be $(c = E, w_i = \text{the})$, which returns the value 1 when $c = E$ and the current word is 'the', and returns 0 otherwise. Given a d-dimensional parameter vector ϕ, the output of the classifier is that class which maximizes the dot product between the feature and parameter vectors:

$$\hat{c}(i, w_1 \dots w_k) = \underset{c \in C}{\operatorname{argmax}} \ \phi \cdot f(c, i, w_1 \dots w_k) \quad (1)$$

In training, the weights in ϕ are initialized to 0. For m epochs (passes over the training data), for each word in the training data (except sentence final words), the model is updated. Let i be the current word position in string $w_1 \dots w_k$ and suppose that there have been $j-1$ previous updates to the model parameters. Let $\bar{c}_i$ be the true class label, and let $\hat{c}_i$ be shorthand for $\hat{c}(i, w_1 \dots w_k)$ in equation 1. Then the parameter vector ϕ_j at step j is updated as follows:

$$\phi_j = \phi_{j-1} - f(\hat{c}, i, w_1 \dots w_k) + f(\bar{c}, i, w_1 \dots w_k) \quad (2)$$

As stated in Section 2.1, we reserved every tenth sentence as held-out data. After each pass over the training data, we evaluated the system performance on this held-out data, and chose the model that optimized accuracy on that set. The averaged perceptron was used on held-out and evaluation sets. See Collins (2002) for more details on this approach.

2.5 Features

To tease apart the utility of finite-state derived features and context-free derived features, we consider three feature sets: (1) basic finite-state features; (2) context-free features; and (3) finite-state approximation to context-free features. Note that every feature must include exactly one class label c in order to discriminate between classes in equation 1. Hence when presenting features, it can be assumed that the class label is part of the feature, even if it is not explicitly mentioned.

The three feature sets are not completely disjoint. We include simple position-based features in every system, defined as follows. Because *edus* are typically multi-word strings, it is less likely for a word near the beginning or end of a sentence to be at an *edu* boundary. Thus it is reasonable to expect the position within a sentence of a token to be a helpful feature. We created 101 indicator features, representing percentages from 0 to 100. For a string of length k, at position i, we round i/k to two decimal places and provide a value of 1 for the corresponding quantized position feature and 0 for the other position features.

2.5.1 Basic finite-state features

Our baseline finite-state feature set includes simple tagger derived features, as well as features based on position in the string and n-grams[4]. We annotate tag sequences onto the word sequence via a competitive discriminatively trained tagger (Hollingshead et al., 2005), trained for each of two kinds of tag sequences: part-of-speech (POS) tags and shallow parse tags. The shallow parse tags define non-hierarchical base constituents ("chunks"), as defined for the CoNLL-2000 shared task (Tjong Kim Sang and Buchholz, 2000). These can either be used as tag or chunk sequences. For example, the tree in Figure 2 represents a shallow (non-hierarchical) parse tree, with four base constituents. Each base constituent X begins with a word labeled with B_X, which signifies that this word begins the constituent. All other words within a constituent X are labeled

[3] Because of the sparsity of boundary tags, Markov dependencies between tags buy no additional system accuracy.

[4] We tried using a list of 311 cue phrases from Knott (1996) to define features, but did not derive any system improvement through this list, presumably because our simple n-gram features already capture many such lexical cues.

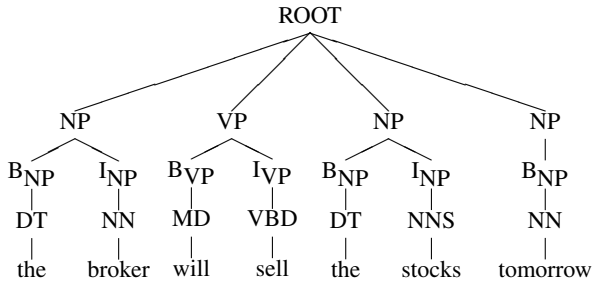


Figure 2: Tree representation of shallow parses, with B(egin) and I(nside) tags

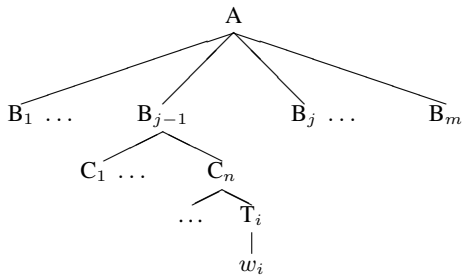


Figure 3: Parse tree schematic for describing context-free segmentation features

I_X, and words outside of any base constituent are labeled O. In such a way, each word is labeled with both a POS-tag and a B/I/O tag.

For our three sequences (lexical, POS-tag and shallow tag), we define n-gram features surrounding the potential discourse boundary. If the current word is w_i, the hypothesized boundary will occur between w_i and w_{i+1}. For this boundary position, the 6-gram including the three words before and the three words after the boundary is included as a feature; additionally, all n-grams for $n < 6$ such that either w_i or w_{i+1} (or both) is in the n-gram are included as features. In other words, all n-grams in a six word window of boundary position i are included as features, except those that include neither w_i nor w_{i+1} in the n-gram. The identical feature templates are used with POS-tag and shallow tag sequences as well, to define tag n-gram features.

This feature set is very close to that used in Sporleder and Lapata (2005), but not identical. Their n-gram feature definitions were different (though similar), and they made use of cue phrases from Knott (1996). In addition, they used a rule-based clauser that we did not. Despite such differences, this feature set is quite close to what is described in that paper.

2.5.2 Context-free features

To describe our context-free features, we first present how SPADE made use of context-free parse trees within their segmentation algorithm, since this forms the basis of our features. The SPADE features are based on productions extracted from full syntactic parses of the given sentence. The primary feature for a discourse boundary after word w_i is based on the lowest constituent in the tree that spans words $w_m \dots w_n$ such that $m \leq i < n$. For example, in the parse tree schematic in Figure 3, the constituent labeled with A is the lowest constituent in the tree whose span crosses the potential discourse boundary after w_i. The primary feature is the production that expands this constituent in the tree, with the proposed segmentation boundary marked, which in this case is: $A \rightarrow B_1 \dots B_{j-1} || B_j \dots B_m$, where $||$ denotes the segmentation boundary. In SPADE, the production is lexicalized by the head words of each constituent, which are determined using standard head-percolation techniques. This feature is used to predict a boundary as follows: if the relative frequency estimate of a boundary given the production feature in the corpus is greater than 0.5, then a boundary is predicted; otherwise not. If the production has not been observed frequently enough, the lexicalization is removed and the relative frequency of a boundary given the unlexicalized production is used for prediction. If the observations of the unlexicalized production are also too sparse, then only the children adjacent to the boundary are maintained in the feature, e.g., $A \rightarrow *B_{j-1} || B_j *$ where $*$ represents zero or more categories. Further smoothing is used when even this is unobserved.

We use these features as the starting point for our context-free feature set: the lexicalized production $A \rightarrow B_1 \dots B_{j-1} || B_j \dots B_m$, as defined above for SPADE, is a feature in our model, as is the unlexicalized version of the production. As with the other features that we have described, this feature is used as an indicator feature in the classifier applied at the word w_i preceding the hypothesized boundary. In addition to these full production features, we use the production with only children adjacent to the boundary, denoted by $A \rightarrow *B_{j-1} || B_j *$. This production is used in four ways: fully lexicalized; unlexicalized; only category B_{j-1} lexicalized; and only category B_j lexicalized. We also use $A \rightarrow *B_{j-2} B_{j-1} || *$ and $A \rightarrow * || B_j B_{j+1} *$ features, both unlexicalized and with the boundary-adjacent category lexicalized. If there is no category B_{j-2} or B_{j+1}, they are replaced with "N/A".

In addition to these features, we fire the same features for all productions on the path from A down

Segmentation system	Segment Boundary accuracy			Bracketing accuracy		
	Recall	Precision	F1	Recall	Precision	F1
SPADE	85.4	85.5	85.5	77.7	77.9	77.8
Classifier: Basic finite-state	81.5	83.3	82.4	73.6	74.5	74.0
Classifier: Full finite-state	84.1	87.9	86.0	78.0	80.0	79.0
Classifier: Context-free	84.7	91.1	87.8	80.3	83.7	82.0
Classifier: All features	89.7	91.3	90.5	84.9	85.8	85.3

Table 2: Segmentation results on all 991 sentences in the RST-DT test set. Segment boundary accuracy is for sentence internal boundaries only, following Soricut and Marcu (2003). Bracketing accuracy is for unlabeled flat bracketing of the same segments. While boundary accuracy correctly depicts segmentation results, the harsher flat bracketing metric better predicts discourse parsing performance.

to the word w_i. For these productions, the segmentation boundary || will occur after all children in the production, e.g., $B_{j-1} \rightarrow C_1 \ldots C_n||$, which is then used in both lexicalized and unlexicalized forms. For the feature with only categories adjacent to the boundary, we again use "N/A" to denote the fact that no category occurs to the right of the boundary: $B_{j-1} \rightarrow *C_n||$N/A. Once again, these are lexicalized as described above.

2.5.3 Finite-state approximation features

An approximation to our context-free features can be made by using the shallow parse tree, as shown in Figure 2, in lieu of the full hierarchical parse tree. For example, if the current word was "sell" in the tree in Figure 2, the primary feature would be ROOT $\rightarrow$ NP VP||NP NP, and it would have an unlexicalized version and three lexicalized versions: the category immediately prior to the boundary lexicalized; the category immediately after the boundary lexicalized; and both lexicalized. For lexicalization, we choose the final word in the constituent as the lexical head for the constituent. This is a reasonable first approximation, because such typically left-headed categories as PP and VP lose their arguments in the shallow parse.

As a practical matter, we limit the number of categories in the flat production to 8 to the left and 8 to the right of the boundary. In a manner similar to the n-gram features that we defined in Section 2.5.1, we allow all combinations with less than 8 contiguous categories on each side, provided that at least one of the adjacent categories is included in the feature. Each feature has an unlexicalized and three lexicalized versions, as described above.

3 Experiments

We performed a number of experiments to determine the relative utility of features derived from full context-free syntactic parses and those derived solely from shallow finite-state tagging. Our primary concern is with intra-sentential discourse segmentation, but we are also interested in how much the improved segmentation helps discourse parsing.

The syntactic parser we use for all context-free syntactic parses used in either SPADE or our classifier is the Charniak parser with reranking, as described in Charniak and Johnson (2005). The Charniak parser and reranker were trained on the sections of the Penn Treebank not included in the RST-DT test set.

All statistical significance testing is done via the stratified shuffling test (Yeh, 2000).

3.1 Segmentation

Table 2 presents segmentation results for SPADE and four versions of our classifier. The "Basic finite-state" system uses only finite-state sequence features as defined in Section 2.5.1, while the "Full finite-state" also includes the finite-state approximation features from Section 2.5.3. The "Context-free" system uses the SPADE-inspired features detailed in Section 2.5.2, but none of the features from Sections 2.5.1 or 2.5.3. Finally, the "All features" section includes features from all three sections.[5]

Note that the full finite-state system is considerably better than the basic finite-state system, demonstrating the utility of these approximations of the SPADE-like context-free features. The performance of the resulting "Full" finite-state system is not statistically significantly different from that of SPADE (p=0.193), despite no reliance on features derived from context-free parses.

The context-free features, however, even without any of the finite-state sequence features (even lexical n-grams) outperforms the best finite-state system by almost two percent absolute, and the system with all features improves on the best finite-state system by over four percent absolute. The system

[5] In the "All features" condition, the finite-state approximation features defined in Section 2.5.3 only include a maximum of 3 children to the left and right of the boundary, versus a maximum of 8 for the "Full finite-state" system. This was found to be optimal on the development set.

Segmentation	Unlabeled	Nuc/Sat
SPADE	76.9	70.2
Classifier: Full finite state	78.1	71.1
Classifier: All features	83.5	76.1

Table 3: Discourse parsing results on the 951 sentence Soricut and Marcu (2003) evaluation set, using SPADE for parsing, and various methods for segmentation. Scores are unlabeled and labeled (Nucleus/Satellite) bracketing accuracy (F1). The first line shows SPADE performing both segmentation and discourse parsing. The other two lines show SPADE performing discourse parsing with segmentations produced by our classifier using different combinations of features.

with all features is statistically significantly better than both SPADE and the "Full finite-state" classifier system, at $p < 0.001$. This large improvement demonstrates that the context-free features can provide a very large system improvement.

3.2 Discourse parsing

It has been shown that accurate discourse segmentation within a sentence greatly improves the overall parsing accuracy to near human levels (Soricut and Marcu, 2003). Given our improved segmentation results presented in the previous section, improvements would be expected in full sentence-level discourse parsing. To achieve this, we modified the SPADE script to accept our segmentations when building the fully hierarchical discourse tree. The results for three systems are presented in Table 3: SPADE, our "Full finite-state" system, and our system with all features. Results for unlabeled bracketing are presented, along with results for labeled bracketing, where the label is either Nucleus or Satellite, depending upon whether or not the node is more central (Nucleus) to the coherence of the text than its sibling(s) (Satellite). This label set has been shown to be of particular utility for indicating which segments are more important to include in an automatically created summary or compressed sentence (Sporleder and Lapata, 2005; Marcu, 1998; Marcu, 1999; Cristea et al., 2005).

Once again, the finite-state system does not perform statistically significantly different from SPADE on either labeled or unlabeled discourse parsing. Using all features, however, results in greater than 5% absolute accuracy improvement over both of these systems, which is significant, in all cases, at $p < 0.001$.

4 Discussion and future directions

Our results show that context-free parse derived features are critical for achieving the highest level of accuracy in sentence-level discourse segmentation. Given that *edus* are by definition clause-like units, it is not surprising that accurate full syntactic parse trees provide highly relevant information unavailable from finite-state approaches. Adding context-free features to our best finite-state feature model reduces error in segmentation by 32.1%, an increase in absolute F-score of 4.5%. These increases are against a finite-state segmentation model that is powerful enough to be statistically indistinguishable from SPADE.

Our experiments also confirm that increased segmentation accuracy yields significantly better discourse parsing accuracy, as previously shown to be the case when providing reference segmentations to a parser (Soricut and Marcu, 2003). The segmentation reduction in error of 34.5% propagates to a 28.6% reduction in error for unlabeled discourse parse trees, and a 19.8% reduction in error for trees labeled with Nucleus and Satellite.

We have several key directions in which to continue this work. First, given that a general machine learning approach allowed us to improve upon SPADE's segmentation performance, we also believe that it will prove useful for improving full discourse parsing, both at the sentence level and beyond. For efficient inter-sentential discourse parsing, we see the need for an additional level of segmentation at the paragraph level. Whereas most sentences correspond to a well-formed subtree, Sporleder and Lascarides (2004) report that over 20% of the paragraph boundaries in the RST-DT do not correspond to a well-formed subtree in the human annotated discourse parse for that document. Therefore, to perform accurate and efficient parsing of the RST-DT at the paragraph level, the text should be segmented into paragraph-like segments that conform to the human-annotated subtree boundaries, just as sentences are segmented into *edus*.

We also intend to begin work on the other discourse annotated corpora. While most work on textual discourse parsing has made use of the RST-DT corpus, the Discourse GraphBank corpus provides a competing annotation that is not constrained to tree structures (Wolf and Gibson, 2005). Once accurate levels of segmentation and parsing for both corpora are attained, it will be possible to perform extrinsic evaluations to determine their relative utility for different NLP tasks. Recent work has shown promising preliminary results for recognizing and labeling relations of GraphBank structures (Wellner et al., 2006), in the case that the algorithm is provided with

manually segmented sentences. Sentence-level segmentation in the GraphBank is very similar to that in the RST-DT, so our segmentation approach should work well for Discourse GraphBank style parsing.

The Penn Discourse Treebank (Miltsakaki et al., 2004), or PDTB, uses a relatively flat annotation of discourse structure, in contrast to the hierarchical structures found in the other two corpora. It contains annotations for discourse connectives and their arguments, where an argument can be as small as a nominalization or as large as several sentences. This approach obviates the need to create a set of discourse relations, but sentence internal segmentation is still a necessary step. Though segmentation in the PDTB tends to larger units than *edus*, our approach to segmentation should be straightforwardly applicable to their segmentation style.

Acknowledgments

Thanks to Caroline Sporleder and Mirella Lapata for their test data and helpful comments. Thanks also to Radu Soricut for helpful input. This research was supported in part by NSF Grant #IIS-0447214. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF.

References

E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M.P. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of english grammars. In *DARPA Speech and Natural Language Workshop*, pages 306–311.

L. Carlson, D. Marcu, and M.E. Okurowski. 2002. RST discourse treebank. Linguistic Data Consortium, Catalog # LDC2002T07. ISBN LDC2002T07.

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

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics*, pages 132–139.

M.J. Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–8.

D. Cristea, O. Postolache, and I. Pistol. 2005. Summarisation through discourse structure. In *6th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing)*.

K. Hollingshead, S. Fisher, and B. Roark. 2005. Comparing and combining finite-state and context-free parsers. In *Proceedings of HLT-EMNLP*, pages 787–794.

A. Knott. 1996. *A Data-Driven Methodology for Motivating a Set of Coherence Relations*. Ph.D. thesis, Department of Artificial Intelligence, University of Edinburgh.

W.C. Mann and S.A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

D. Marcu. 1998. Improving summarization through rhetorical parsing tuning. In *The 6th Workshop on Very Large Corpora*.

D. Marcu. 1999. Discourse trees are good indicators of importance in text. In I. Mani and M. Maybury, editors, *Advances in Automatic Text Summarization*, pages 123–136. MIT Press, Cambridge, MA.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

E. Miltsakaki, R. Prasad, A. Joshi, and B. Webber. 2004. The Penn Discourse TreeBank. In *Proceedings of the Language Resources and Evaluation Conference*.

R. Prasad, A. Joshi, N. Dinesh, A. Lee, E. Miltsakaki, and B. Webber. 2005. The Penn Discourse TreeBank as a resource for natural language generation. In *Proceedings of the Corpus Linguistics Workshop on Using Corpora for Natural Language Generation*.

R. Soricut and D. Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Human Language Technology Conference of the North American Association for Computational Linguistics (HLT-NAACL)*.

C. Sporleder and M. Lapata. 2005. Discourse chunking and its application to sentence compression. In *Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 257–264.

C. Sporleder and A. Lascarides. 2004. Combining hierarchical clustering and machine learning to predict high-level discourse structure. In *Proceedings of the International Conference in Computational Linguistics (COLING)*, pages 43–49.

E.F. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL*, pages 127–132.

S. Verberne, L. Boves, N. Oostdijk, and P.A. Coppen. 2006. Discourse-based answering of why-questions. *Traitement Automatique des Langues (TAL)*.

B. Wellner, J. Pustejovsky, C. Havasi, A. Rumshisky, and R. Sauri. 2006. Classification of discourse coherence relations: An exploratory study using multiple knowledge sources. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*.

F. Wolf and E. Gibson. 2005. Representing discourse coherence: A corpus-based analysis. *Computational Linguistics*, 31(2):249–288.

A. Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International COLING*, pages 947–953.

PERSONAGE: Personality Generation for Dialogue

François Mairesse
Department of Computer Science
University of Sheffield
Sheffield, S1 4DP, United Kingdom
F.Mairesse@sheffield.ac.uk

Marilyn Walker
Department of Computer Science
University of Sheffield
Sheffield, S1 4DP, United Kingdom
M.A.Walker@sheffield.ac.uk

Abstract

Over the last fifty years, the "Big Five" model of personality traits has become a standard in psychology, and research has systematically documented correlations between a wide range of linguistic variables and the Big Five traits. A distinct line of research has explored methods for automatically generating language that varies along personality dimensions. We present PERSONAGE (PERSONAlity GEnerator), the first highly parametrizable language generator for extraversion, an important aspect of personality. We evaluate two personality generation methods: (1) direct generation with particular parameter settings suggested by the psychology literature; and (2) overgeneration and selection using statistical models trained from judge's ratings. Results show that both methods reliably generate utterances that vary along the extraversion dimension, according to human judges.

1 Introduction

Over the last fifty years, the "Big Five" model of personality traits has become a standard in psychology (extraversion, neuroticism, agreeableness, conscientiousness, and openness to experience), and research has systematically documented correlations between a wide range of linguistic variables and the Big Five traits (Mehl et al., 2006; Norman, 1963; Oberlander and Gill, 2006; Pennebaker and King, 1999). A distinct line of research has explored methods for automatically generating language that varies along personality dimensions, targeting applications such as computer gaming and educational virtual worlds (André et al., 2000; Isard et al., 2006; Loyall and Bates, 1997; Piwek, 2003; Walker et al., 1997) *inter alia*. Other work suggests a clear utility for generating language manifesting personality (Reeves and Nass, 1996). However, to date, (1) research in generation has not systematically exploited the psycholinguistic findings; and (2) there has been little evaluation showing that automatic generators can produce language with recognizable personality variation.

Alt	Realization	Extra
5	Err... it seems to me that Le Marais isn't as bad as the others.	1.83
4	Right, I mean, Le Marais is the only restaurant that is any good.	2.83
8	Ok, I mean, Le Marais is a quite french, kosher and steak house place, you know and the atmosphere isn't nasty, it has nice atmosphere. It has friendly service. It seems to me that the service is nice. It isn't as bad as the others, is it?	5.17
9	Well, it seems to me that I am sure you would like Le Marais. It has good food, the food is sort of rather tasty, the ambience is nice, the atmosphere isn't sort of nasty, it features rather friendly servers and its price is around 44 dollars.	5.83
3	I am sure you would like Le Marais, you know. The atmosphere is acceptable, the servers are nice and it's a french, kosher and steak house place. Actually, the food is good, even if its price is 44 dollars.	6.00
10	It seems to me that Le Marais isn't as bad as the others. It's a french, kosher and steak house place. It has friendly servers, you know but it's somewhat expensive, you know!	6.17
2	Basically, actually, I am sure you would like Le Marais. It features friendly service and acceptable atmosphere and it's a french, kosher and steak house place. Even if its price is 44 dollars, it just has really good food, nice food.	6.17

Table 1: Recommendations along the extraversion dimension, with the average extraversion rating from human judges on a scale from 1 to 7. Alt-2 and 3 are from the extravert set, Alt-4 and 5 are from the introvert set, and others were randomly generated.

Our aim is to produce a highly parameterizable generator whose outputs vary along personality dimensions. We hypothesize that such language can

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 496–503,
Prague, Czech Republic, June 2007.

be generated by varying parameters suggested by psycholinguistic research. So, we must first map the psychological findings to parameters of a natural language generator (NLG). However, this presents several challenges: (1) The findings result from studies of genres of language, such as stream-of-consciousness essays (Pennebaker and King, 1999), and informal conversations (Mehl et al., 2006), and thus may not apply to fixed content domains used in NLG; (2) Most findings are based on self-reports of personality, but we want to affect observer's perceptions; (3) The findings consist of weak but significant correlations, so that individual parameters may not have a strong enough effect to produce recognizable variation within a single utterance; (4) There are many possible mappings of the findings to generation parameters; and (5) It is unclear whether only specific speech-act types manifest personality or whether all utterances do.

Thus this paper makes several contributions. First, Section 2 summarizes the linguistic reflexes of extraversion, organized by the modules in a standard NLG system, and propose a mapping from these findings to NLG parameters. To our knowledge this is the first attempt to put forward a systematic framework for generating language manifesting personality. We start with the extraversion dimension because it is an important personality factor, with many associated linguistic variables. We believe that our framework will generalize to the other dimensions in the Big Five model. Second, Sections 3 and 4 describe the PERSONAGE (PERSONAlity GEnerator) generator and its 29 parameters. Table 1 shows examples generated by PERSONAGE for recommendations in the restaurant domain, along with human extraversion judgments. Third, Sections 5 and 6 describe experiments evaluating two generation methods. We first show that (1) the parameters generate utterances that vary significantly on the extraversion dimension, according to human judgments; and (2) we can train a statistical model that matches human performance in assigning extraversion ratings to generation outputs produced with random parameter settings. Section 7 sums up and discusses future work.

2 Psycholinguistic Findings and PERSONAGE Parameters

We hypothesize that personality can be made manifest in evaluative speech acts in any dialogue domain, i.e. utterances responding to requests to RECOMMEND or COMPARE domain entities, such as restaurants or movies (Isard et al., 2006; Stent et al., 2004). Thus, we start with the SPaRKy generator[1], which produces evaluative recommendations and comparisons in the restaurant domain, for a database of restaurants in New York City. There are eight attributes for each restaurant: the name and address, scalar attributes for *price*, *food quality*, *atmosphere*, and *service* and categorical attributes for *neighborhood* and *type of cuisine*. SPaRKy is based on the standard NLG architecture (Reiter and Dale, 2000), and consists of the following modules:

1. Content Planning: refine communicative goals, select and structure content;
2. Sentence planning; choose linguistic resources (lexicon, syntax) to achieve goals;
3. Realization: use grammar (syntax, morphology) to generate surface utterances.

Given the NLG architecture, speech-act types, and domain, the first step then is to summarise psychological findings on extraversion and map them to this architecture. The column **NLG modules** of Table 2 gives the proposed mapping. The first row specifies findings for the content planning module and the other rows are aspects of sentence planning. Realization is achieved with the RealPro surface realizer (Lavoie and Rambow, 1997). An examination of the introvert and extravert findings in Table 2 highlights the challenges above, i.e. exploiting these findings in a systematic way within a parameterizable NLG system.

The column **Parameter** in Table 2 proposes parameters (explained in Sections 3 and 4) that are manipulated within each module to realize the findings in the other columns. Each parameter varies continuously from 0 to 1, where end points are meant to produce extreme but plausible output. Given the challenges above, it is important to note that these parameters represent *hypotheses* about how a finding can be mapped into *any* NLG system. The **Intro** and **Extra** columns at the right hand side of the **Parameter** column indicate a range of settings for this parameter, suggested by the psychological findings, to produce introverted vs. extraverted language.

SPaRKy produces content plans for restaurant recommendations and comparisons that are modified by the parameters. The sample content plan for a recommendation in Figure 1 corresponds to the outputs in Table 1. While Table 1 shows that PERSONAGE's parameters have various pragmatic effects, they preserve the meaning at the Gricean intention level (dialogue goal). Each content plan contains a claim (nucleus) about the overall quality of

[1] Available for download from `www.dcs.shef.ac.uk/cogsys/sparky.html`

NLG modules	Introvert findings	Extravert findings	Parameter	Intro	Extra
Content selection and structure	Single topic	Many topics	VERBOSITY	low	high
	Strict selection	Think out loud*	RESTATEMENTS	low	high
			REPETITIONS	low	low
	Problem talk, dissatisfaction	Pleasure talk, agreement, compliment	CONTENT POLARITY	low	high
			REPETITIONS POLARITY	low	high
			CLAIM POLARITY	low	high
			CONCESSIONS	avg	avg
			CONCESSIONS POLARITY	low	high
			POLARISATION	low	high
			POSITIVE CONTENT FIRST	low	high
Syntactic templates selection	Few self-references	Many self-references	SELF-REFERENCES	low	high
	Elaborated constructions	Simple constructions*	CLAIM COMPLEXITY	high	low
	Many articles	Few articles			
Aggregation Operations	Many words per sentence/clause	Few words per sentence/clause	RELATIVE CLAUSES	high	low
			WITH CUE WORD	high	low
			CONJUNCTION	low	high
	Many unfilled pauses	Few unfilled pauses	PERIOD	high	low
			...		
Pragmatic transformations	Many nouns, adjectives, prepositions (explicit)	Many verbs, adverbs, pronouns (implicit)	SUBJECT IMPLICITNESS	low	high
	Many negations	Few negations	NEGATION INSERTION	high	low
	Many tentative words	Few tentative words	DOWNTONER HEDGES:		
			·SORT OF, SOMEWHAT, QUITE, RATHER, ERR, I THINK THAT, IT SEEMS THAT, IT SEEMS TO ME THAT, I MEAN	high	low
			·AROUND	avg	avg
	Formal	Informal	·KIND OF, LIKE	low	high
			ACKNOWLEDGMENTS:		
			·YEAH	low	high
			·RIGHT, OK, I SEE, WELL	high	low
	Realism	Exaggeration*	EMPHASIZER HEDGES:		
			·REALLY, BASICALLY, ACTUALLY, JUST HAVE, JUST IS, EXCLAMATION	low	high
			·YOU KNOW	low	high
	No politeness form	Positive face redressment*	TAG QUESTION INSERTION	low	high
	Lower word count	Higher word count	HEDGE VARIATION	low	avg
			HEDGE REPETITION	low	low
Lexical choice	Rich	Poor	LEXICON FREQUENCY	low	high
	Few positive emotion words	Many positive emotion words	*see polarity parameters*		
	Many negative emotion words	Few negative emotion words	*see polarity parameters*		

Table 2: Summary of language cues for extraversion, based on Dewaele and Furnham (1999); Furnham (1990); Mehl et al. (2006); Oberlander and Gill (2006); Pennebaker and King (1999), as well as PERSONAGE's corresponding generation parameters. Asterisks indicate hypotheses, rather than results. For details on aggregation parameters, see Section 4.2.

Relations:	JUSTIFY (nuc:1, sat:2); JUSTIFY (nuc:1, sat:3); JUSTIFY (nuc:1, sat:4); JUSTIFY (nuc:1, sat:5); JUSTIFY (nuc:1, sat:6)
Content:	1. assert(best (*Le Marais*)) 2. assert(is (*Le Marais*, cuisine (*French*))) 3. assert(has (*Le Marais*, food-quality (*good*))) 4. assert(has (*Le Marais*, service (*good*))) 5. assert(has (*Le Marais*, decor (*decent*))) 6. assert(is (*Le Marais*, price (*44 dollars*)))

Figure 1: A content plan for a recommendation.

the selected restaurant(s), supported by a set of satellite content items describing their attributes. See Table 1. Claims can be expressed in different ways, such as *RESTAURANT_NAME is the best*, while the attribute satellites follow the pattern *RESTAURANT_NAME has MODIFIER ATTRIBUTE_NAME*, as in *Le Marais has good food*. Recommendations are characterized by a JUSTIFY rhetorical relation associating the claim with all other content items, which are linked together through an INFER relation. In comparisons, the attributes of multiple restaurants are compared using a CONTRAST relation. An optional claim about the quality of all restaurants can also be expressed as the nucleus of an ELABORATE relation, with the rest of the content plan tree as a satellite.

3 Content Planning

Content planning selects and structures the content to be communicated. Table 2 specifies 10 parameters hypothesized to affect this process which are explained below.

Content size: Extraverts are more talkative than introverts (Furnham, 1990; Pennebaker and King, 1999), although it is not clear whether they actually produce more content, or are just redundant and wordy. Thus various parameters relate to the amount and type of content produced. The VERBOSITY parameter controls the number of content items selected from the content plan. For example, Alt-5 in Table 1 is terse, while Alt-2 expresses all the items in the content plan. The REPETITION parameter adds an exact repetition: the content item is duplicated and linked to the original content by a RESTATE

rhetorical relation. In a similar way, the RESTATEMENT parameter adds paraphrases of content items to the plan, that are obtained from the initial handcrafted generation dictionary (see Section 4.1) and by automatically substituting content words with the most frequent WordNet synonym (see Section 4.4). Alt-9 in Table 1 contains restatements for the food quality and the atmosphere attributes.

Polarity: Extraverts tend to be more positive; introverts are characterized as engaging in more 'problem talk' and expressions of dissatisfaction (Thorne, 1987). To control for polarity, content items are defined as positive or negative based on the scalar value of the corresponding attribute. The *type of cuisine* and *neighborhood* attributes have neutral polarity. There are multiple parameters associated with polarity. The CONTENT POLARITY parameter controls whether the content is mostly negative (e.g. *X has mediocre food*), neutral (e.g. *X is a Thai restaurant*), or positive. From the filtered set of content items, the POLARISATION parameter determines whether the final content includes items with extreme scalar values (e.g. *X has fantastic staff*).

In addition, polarity can also be implied more subtly through rhetorical structure. The CONCESSIONS parameter controls how negative and positive information is presented, i.e. whether two content items with different polarity are presented objectively, or if one is foregrounded and the other backgrounded. If two opposed content items are selected for a concession, a CONCESS rhetorical relation is inserted between them. While the CONCESSIONS parameter captures the tendency to put information into perspective, the CONCESSION POLARITY parameter controls whether the positive or the negative content is concessed, i.e. marked as the satellite of the CONCESS relation. The last sentence of Alt-3 in Table 1 illustrates a positive concession, in which the good food quality is put before the high price.

Content ordering: Although extraverts use more positive language (Pennebaker and King, 1999; Thorne, 1987), it is unclear how they position the positive content within their utterances. Additionally, the position of the claim affects the persuasiveness of an argument (Carenini and Moore, 2000): starting with the claim facilitates the hearer's understanding, while finishing with the claim is more effective if the hearer disagrees. The POSITIVE CONTENT FIRST parameter therefore controls whether positive content items – including the claim – appear first or last, and the order in which the content items are aggregated. However, some operations can still impose a specific ordering (e.g. BECAUSE cue word to realize the JUSTIFY relation, see Section 4.2).

4 Sentence Planning

Sentence planning chooses the linguistic resources from the lexicon and the syntactic and discourse structures to achieve the communicative goals specified in the input content plan. Table 2 specifies four sets of findings and parameters for different aspects of sentence planning discussed below.

4.1 Syntactic template selection

PERSONAGE's input generation dictionary is made of 27 Deep Syntactic Structures (DSyntS): 9 for the recommendation claim, 12 for the comparison claim, and one per attribute. Selecting a DSyntS requires assigning it automatically to a point in a three dimensional space described below. All parameter values are normalized over all the DSyntS, so the DSyntS closest to the target value can be computed.

Syntactic complexity: Furnham (1990) suggests that introverts produce more complex constructions: the CLAIM COMPLEXITY parameter controls the depth of the syntactic structure chosen to represent the claim, e.g. the claim *X is the best* is rated as less complex than *X is one of my favorite restaurants*.

Self-references: Extraverts make more self-references than introverts (Pennebaker and King, 1999). The SELF-REFERENCE parameter controls whether the claim is made in the first person, based on the speaker's own experience, or whether the claim is reported as objective or information obtained elsewhere. The self-reference value is obtained from the syntactic structure by counting the number of first person pronouns. For example, the claim of Alt-2 in Table 1, i.e. *I am sure you would like Le Marais*, will be rated higher than *Le Marais isn't as bad as the others* in Alt-5.

Polarity: While polarity can be expressed by content selection and structure, it can also be directly associated with the DSyntS. The CLAIM POLARITY parameter determines the DSyntS selected to realize the claim. DSyntS are manually annotated for polarity. For example, Alt-4's claim in Table 1, i.e. *Le Marais is the only restaurant that is any good*, has a lower polarity than Alt-2.

4.2 Aggregation operations

SPaRKy aggregation operations are used (See Stent et al. (2004)), with additional operations for concessions and restatements. See Table 2. The probability of the operations biases the production of complex clauses, periods and formal cue words for introverts, to express their preference for complex syn-

tactic constructions, long pauses and rich vocabulary (Furnham, 1990). Thus, the introvert parameters favor operations such as RELATIVE CLAUSE for the INFER relation, PERIOD HOWEVER CUE WORD for CONTRAST, and ALTHOUGH ADVERBIAL CLAUSE for CONCESS, that we hypothesize to result in more formal language. Extravert aggregation produces longer sentences with simpler constructions and informal cue words. Thus extravert utterances tend to use operations such as a CONJUNCTION to realize the INFER and RESTATE relations, and the EVEN IF ADVERBIAL CLAUSE for CONCESS relations.

4.3 Pragmatic transformations

This section describes the insertion of markers in the DSyntS to produce various pragmatic effects.

Hedges: Hedges correlate with introversion (Pennebaker and King, 1999) and affect politeness (Brown and Levinson, 1987). Thus there are parameters for inserting a wide range of hedges, both affective and epistemic, such as *kind of, sort of, quite, rather, somewhat, like, around, err, I think that, it seems that, it seems to me that,* and *I mean*. Alt-5 in Table 1 shows hedges *err* and *it seems to me that*.

To model extraverts use of more social language, agreement and backchannel behavior (Dewaele and Furnham, 1999; Pennebaker and King, 1999), we use informal acknowledgments such as *yeah, right, ok*. Acknowledgments that may affect introversion are *I see*, expressing self-reference and cognitive load, and the *well* cue word implying reservation from the speaker (see Alt-9).

To model social connection and emotion we added mechanisms for inserting emphasizers such as *you know, basically, actually, just have, just is,* and exclamations. Alt-3 in Table 1 shows the insertion of *you know* and *actually*.

Although similar hedges can be grouped together, each hedge has a unique pragmatic effect. For example, *you know* implies positive-face redressment, while *actually* doesn't. A parameter for each hedge controls the likelihood of its selection.

To control the general level of hedging, a HEDGE VARIATION parameter defines how many different hedges are selected (maximum of 5), while the frequency of an individual hedge is controlled by a HEDGE REPETITION parameter, up to a maximum of 2 identical hedges per utterance.

The syntactic structure of hedges are defined as well as constraints on their insertion point in the utterance's syntactic structure. Each time a hedge is selected, it is randomly inserted at one of the insertion points respecting the constraints, until the specified frequency is reached. For example, a constraint on the hedge *kind of* is that it modifies adjectives.

Tag questions: Tag questions are also politeness markers (Brown and Levinson, 1987). They redress the hearer's positive face by claiming common ground. A TAG QUESTION INSERTION parameter leads to negating the auxiliary of the verb and pronominalizing the subject, e.g. *X has great food* results in the insertion of *doesn't it?*, as in Alt-8.

Negations: Introverts use significantly more negations (Pennebaker and King, 1999). Although the content parameters select more negative polarity content items for introvert utterances, we also manipulate negations, while keeping the content constant, by converting adjectives to the negative of their antonyms, e.g. *the atmosphere is nice* was transformed to *not nasty* in Alt-9 in Table 1.

Subject implicitness: Heylighen and Dewaele (2002) found that extraverts use more implicit language than introverts. To control the level of implicitness, the SUBJECT IMPLICITNESS parameter determines whether predicates describing restaurant attributes are expressed with the restaurant in the subject, or with the attribute itself (e.g., *it has good food* vs. *the food is tasty* in Alt-9).

4.4 Lexical choice

Introverts use a richer vocabulary (Dewaele and Furnham, 1999), so the LEXICON FREQUENCY parameter selects lexical items by their normalized frequency in the British National Corpus. WordNet synonyms are used to obtain a pool of synonyms, as well as adjectives extracted from a corpus of restaurant reviews for all levels of polarity (e.g. the adjective *tasty* in Alt-9 is a high polarity modifier of the food attribute). Synonyms are manually checked to make sure they are interchangeable. For example, the content item expressed originally as *it has decent service* is transformed to *it features friendly service* in Alt-2, and to *the servers are nice* in Alt-3.

5 Experimental Method and Hypotheses

Our primary hypothesis is that language generated by varying parameters suggested by psycholinguistic research can be recognized as extravert or introvert. To test this hypothesis, three expert judges evaluated a set of generated utterances as if they had been uttered by a friend responding in a dialogue to a request to recommend restaurants. These utterances had been generated to systematically manipulate extraversion/introversion parameters.

The judges rated each utterance for perceived extraversion, by answering the two questions measur-

ing that trait from the Ten-Item Personality Inventory, as this instrument was shown to be psychometrically superior to a 'single item per trait' questionnaire (Gosling et al., 2003). The answers are averaged to produce an extraversion rating ranging from 1 (highly introvert) to 7 (highly extravert). Because it was unclear whether the generation parameters in Table 2 would produce natural sounding utterances, the judges also evaluated the naturalness of each utterance on the same scale. The judges rated 240 utterances, grouped into 20 sets of 12 utterances generated from the same content plan. They rated one randomly ordered set at a time, but viewed all 12 utterances in that set before rating them. The utterances were generated to meet two experimental goals. First, to test the direct control of the perception of extraversion. 2 introvert utterances and 2 extravert utterances were generated for each content plan (80 in total) using the parameter values in Table 2. Multiple outputs were generated with both parameter settings normally distributed with a 15% standard deviation. Second, 8 utterances for each content plan (160 in total) were generated with random parameter values. These random utterances make it possible to: (1) improve PERSONAGE's direct output by calibrating its parameters more precisely; and (2) build a statistical model that selects utterances matching input personality values after an overgeneration phase (see Section 6.2). The inter-rater agreement for extraversion between the judges over all 240 utterances (average Pearson's correlation of 0.57) shows that the magnitude of the differences of perception between judges is almost constant ($\sigma = .037$). A low agreement can yield a high correlation (e.g. if all values differ by a constant factor), so we also compute the intraclass correlation coefficient r based on a two-way random effect model. We obtain a r of 0.79, which is significant at the $p < .001$ level (reliability of average measures, identical to Cronbach's alpha). This is comparable to the agreement of judgments of personality in Mehl et al. (2006) (mean $r = 0.84$).

6 Experimental Results

6.1 Hypothesized parameter settings

Table 1 provides examples of PERSONAGE's output and extraversion ratings. To assess whether PERSONAGE generates language that can be recognized as introvert and extravert, we did a independent sample t-test between the average ratings of the 40 introvert and 40 extravert utterances (parameters with 15% standard deviation as in Table 2). Table 3 shows that introvert utterances have an average rating of 2.96 out of 7 while extravert utterances have an average rating of 5.98. These ratings are significantly different at the $p < .001$ level (two-tailed). In addition, if we divide the data into two equal-width bins around the neutral extravert rating (4 out of 7), then PERSONAGE's utterance ratings fall in the bin predicted by the parameter set 89.2% of the time. Extravert utterance are also slightly more natural than the introvert ones ($p < .001$).

Rating	Introvert	Extravert	Random
Extraversion	2.96	5.98	5.02
Naturalness	4.93	5.78	4.51

Table 3: Average extraversion and naturalness ratings for the utterances generated with introvert, extravert, and random parameters.

Table 3 also shows that the 160 random parameter utterances produce an average extraversion rating of 5.02, both significantly higher than the introvert set and lower than the extravert set ($p < .001$). Interestingly, the random utterances, which may combine linguistic variables associated with both introverts and extraverts, are less natural than the introvert ($p = .059$) and extravert sets ($p < .001$).

6.2 Statistical models evaluation

We also investigate a second approach: overgeneration with random parameter settings, followed by ranking via a statistical model trained on the judges' feedback. This approach supports generating utterances for any input extraversion value, as well as determining which parameters affect the judges' perception.

We model perceived personality ratings ($1 \ldots 7$) with regression models from the Weka toolbox (Witten and Frank, 2005). We used the full dataset of 160 averaged ratings for the random parameter utterances. Each utterance was associated with a feature vector with the generation decisions for each parameter in Section 2. To reduce data sparsity, we select features that correlate significantly with the ratings ($p < .10$) with a coefficient higher than 0.1.

Regression models are evaluated using the mean absolute error and the correlation between the predicted score and the actual average rating. Table 4 shows the mean absolute error on a scale from 1 to 7 over ten 10-fold cross-validations for the 4 best regression models: Linear Regression (LR), M5' model tree (M5), and Support Vector Machines (i.e. SMOreg) with linear kernels (SMO_1) and radial-

basis function kernels (SMO_r). All models significantly outperform the baseline (0.83 mean absolute error, $p < .05$), but surprisingly the linear model performs the best with a mean absolute error of 0.65. The best model produces a correlation coefficient of 0.59 with the judges' ratings, which is higher than the correlations between pairs of judges, suggesting that the model performs as well as a human judge.

Metric	**LR**	**M5**	**SMO_1**	**SMO_r**
Absolute error	**0.65**	0.66	0.72	0.70
Correlation	**0.59**	0.56	0.54	0.57

Table 4: Mean absolute regression errors (scale from 1 to 7) and correlation coefficients over ten 10-fold cross-validations, for 4 models: Linear Regression (LR), M5' model tree (M5), Support Vector Machines with linear kernels (SMO_1) and radial-basis function kernels (SMO_r). All models significantly outperform the mean baseline (0.83 error, $p < .05$).

The M5' regression tree in Figure 2 assigns a rating given the features. Verbosity plays the most important role: utterances with 4 or more content items are modeled as more extravert. Given a low verbosity, lexical frequency and restatements determine the extraversion level, e.g. utterances with less than 4 content items and infrequent words are perceived as very introverted (rating of 2.69 out of 7). For verbose utterances, the *you know* hedge indicates extraversion, as well as concessions, restatements, self-references, and positive content. Although relatively simple, these models are useful for identifying new personality markers, as well as calibrating parameters in the direct generation model.

7 Discussion and Conclusions

We present and evaluate PERSONAGE, a parameterizable generator that produces outputs that vary along the extraversion personality dimension. This paper makes four contributions:

1. We present a systematic review of psycholinguistic findings, organized by the NLG reference architecture;
2. We propose a mapping from these findings to generation parameters for each NLG module and a real-time implementation of a generator using these parameters[2]. To our knowledge this is the first attempt to put forward a systematic framework for generating language that manifests personality;
3. We present an evaluation experiment showing that we can control the parameters to produce recognizable linguistic variation along the extraversion personality dimension. Thus, we show that the weak correlations reported in other genres of language, and for self-reports rather than observers, carry over to the production of single evaluative utterances with recognizable personality in a restricted domain;
4. We present the results of a training experiment showing that given an output, we can train a model that matches human performance in assigning an extraversion rating to that output.

Some of the challenges discussed in the introduction remain. We have shown that evaluative utterances in the restaurant domain can manifest personality, but more research is needed on which speech acts recognisably manifest personality in a restricted domain. We also showed that the mapping we hypothesised of findings to generation parameters was effective, but there may be additional parameters that the psycholinguistic findings could be mapped to.

Our work was partially inspired by the ICONOCLAST and PAULINE parameterizable generators (Bouayad-Agha et al., 2000; Hovy, 1988), which vary the style, rather than the personality, of the generated texts. Walker et al. (1997) describe a generator intended to affect perceptions of personality, based on Brown and Levinson's theory of politeness (Brown and Levinson, 1987), that uses some of the linguistic constructions implemented here, such as tag questions and hedges, but it was never evaluated. Research by André et al. (2000); Piwek (2003) uses personality variables to affect the linguistic behaviour of conversational agents, but they did not systematically manipulate parameters, and their generators were not evaluated. Reeves and Nass (1996) demonstrate that manipulations of personality affect many aspects of user's perceptions, but their experiments use handcrafted utterances, rather than generated utterances. Cassell and Bickmore (2003) show that extraverts prefer systems utilizing discourse plans that include small talk. Paiva and Evans' trainable generator (2005) produces outputs that correspond to a set of linguistic variables measured in a corpus of target texts. Their method is similar to our statistical method using regression trees, but provides direct control. The method reported in Mairesse and Walker (2005) for training individualized sentence planners ranks the outputs produced by an overgeneration phase, rather than directly predicting a scalar value, as we do here. The closest work to ours is probably Isard et al.'s CRAG-2 system (2006), which overgenerates and ranks using ngram language models trained on a corpus labelled for all Big Five personality dimensions. However, CRAG-2 has no explicit parameter control, and it has yet to be evaluated.

[2] An online demo is available at `www.dcs.shef.ac.uk/cogsys/personage.html`

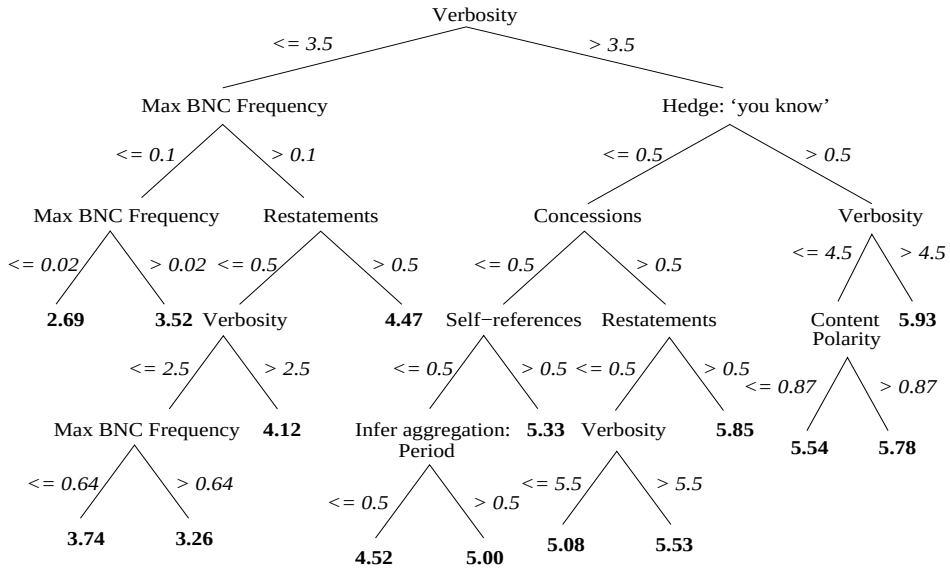


Figure 2: M5' regression tree. The output ranges from 1 to 7, where 7 means strongly extravert.

In future work, we hope to directly compare the direct generation method of Section 6.1 with the overgenerate and rank method of Section 6.2, and to use these results to refine PERSONAGE's parameter settings. We also hope to extend PERSONAGE's generation capabilities to other Big Five traits, identify additional features to improve the model's performance, and evaluate the effect of personality variation on user satisfaction in various applications.

References

E. André, T. Rist, S. van Mulken, M. Klesen, and S. Baldes. 2000. The automated design of believable dialogues for animated presentation teams. In *Embodied conversational agents*, p. 220–255. MIT Press, Cambridge, MA.

N. Bouayad-Agha, D. Scott, and R. Power. 2000. Integrating content and style in documents: a case study of patient information leaflets. *Information Design Journal*, 9:161–176.

P. Brown and S. Levinson. 1987. *Politeness: Some universals in language usage*. Cambridge University Press.

G. Carenini and J. D. Moore. 2000. A strategy for generating evaluative arguments. In *Proc. of International Conference on Natural Language Generation*, p. 47–54.

J. Cassell and T. Bickmore. 2003. Negotiated collusion: Modeling social language and its relationship effects in intelligent agents. *User Modeling and User-Adapted Interaction*, 13 (1-2):89–132.

J-M. Dewaele and A. Furnham. 1999. Extraversion: the unloved variable in applied linguistic research. *Language Learning*, 49(3):509–544.

A. Furnham. 1990. Language and personality. In *Handbook of Language and Social Psychology*. Winley.

S. D. Gosling, P. J. Rentfrow, and W. B. Swann Jr. 2003. A very brief measure of the big five personality domains. *Journal of Research in Personality*, 37:504–528.

F. Heylighen and J-M. Dewaele. 2002. Variation in the contextuality of language: an empirical measure. *Context in Context, Foundations of Science*, 7(3):293–340.

E. Hovy. 1988. *Generating Natural Language under Pragmatic Constraints*. Lawrence Erlbaum Associates.

A. Isard, C. Brockmann, and J. Oberlander. 2006. Individuality and alignment in generated dialogues. In *Proc. of INLG*.

B. Lavoie and O. Rambow. 1997. A fast and portable realizer for text generation systems. In *Proc. of ANLP*.

A. Loyall and J. Bates. 1997. Personality-rich believable agents that use language. In *Proc. of the First International Conference on Autonomous Agents*, p. 106–113.

F. Mairesse and M. Walker. 2005. Learning to personalize spoken generation for dialogue systems. In *Proc. of the Interspeech - Eurospeech*, p. 1881–1884.

M. Mehl, S. Gosling, and J. Pennebaker. 2006. Personality in its natural habitat: Manifestations and implicit folk theories of personality in daily life. *Journal of Personality and Social Psychology*, 90:862–877.

W. T. Norman. 1963. Toward an adequate taxonomy of personality attributes: Replicated factor structure in peer nomination personality rating. *Journal of Abnormal and Social Psychology*, 66:574–583.

J. Oberlander and A. Gill. 2006. Language with character: A stratified corpus comparison of individual differences in e-mail communication. *Discourse Processes*, 42:239–270.

D. Paiva and R. Evans. 2005. Empirically-based control of natural language generation. In *Proc. of ACL*.

J. W. Pennebaker and L. A. King. 1999. Linguistic styles: Language use as an individual difference. *Journal of Personality and Social Psychology*, 77:1296–1312.

P. Piwek. 2003. A flexible pragmatics-driven language generator for animated agents. In *Proc. of EACL*.

B. Reeves and C. Nass. 1996. *The Media Equation*. University of Chicago Press.

E. Reiter and R. Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press.

A. Stent, R. Prasad, and M. Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proc. of ACL*.

A. Thorne. 1987. The press of personality: A study of conversations between introverts and extraverts. *Journal of Personality and Social Psychology*, 53:718–726.

M. Walker, J. Cahn, and S. Whittaker. 1997. Improvising linguistic style: Social and affective bases for agent personality. In *Proc. of the Conference on Autonomous Agents*.

I. H. Witten and E. Frank. 2005. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Making Sense of Sound: Unsupervised Topic Segmentation over Acoustic Input

Igor Malioutov, Alex Park, Regina Barzilay, and **James Glass**
Massachusetts Institute of Technology
{igorm,malex,regina,glass}@csail.mit.edu

Abstract

We address the task of unsupervised topic segmentation of speech data operating over raw acoustic information. In contrast to existing algorithms for topic segmentation of speech, our approach does not require input transcripts. Our method predicts topic changes by analyzing the distribution of reoccurring acoustic patterns in the speech signal corresponding to a single speaker. The algorithm robustly handles noise inherent in acoustic matching by intelligently aggregating information about the similarity profile from multiple local comparisons. Our experiments show that audio-based segmentation compares favorably with transcript-based segmentation computed over noisy transcripts. These results demonstrate the desirability of our method for applications where a speech recognizer is not available, or its output has a high word error rate.

1 Introduction

An important practical application of topic segmentation is the analysis of spoken data. Paragraph breaks, section markers and other structural cues common in written documents are entirely missing in spoken data. Insertion of these structural markers can benefit multiple speech processing applications, including audio browsing, retrieval, and summarization.

Not surprisingly, a variety of methods for topic segmentation have been developed in the past (Beeferman et al., 1999; Galley et al., 2003; Dielmann and Renals, 2005). These methods typically assume that a segmentation algorithm has access not only to acoustic input, but also to its transcript. This assumption is natural for applications where the transcript has to be computed as part of the system output, or it is readily available from other system components. However, for some domains and languages, the transcripts may not be available, or the recognition performance may not be adequate to achieve reliable segmentation. In order to process such data, we need a method for topic segmentation that does not require transcribed input.

In this paper, we explore a method for topic segmentation that operates directly on a raw acoustic speech signal, without using any input transcripts. This method predicts topic changes by analyzing the distribution of reoccurring acoustic patterns in the speech signal corresponding to a single speaker. In the same way that unsupervised segmentation algorithms predict boundaries based on changes in lexical distribution, our algorithm is driven by changes in the distribution of acoustic patterns. The central hypothesis here is that similar sounding acoustic sequences produced by the same speaker correspond to similar lexicographic sequences. Thus, by analyzing the distribution of acoustic patterns we could approximate a traditional content analysis based on the lexical distribution of words in a transcript.

Analyzing high-level content structure based on low-level acoustic features poses interesting computational and linguistic challenges. For instance, we need to handle the noise inherent in matching based on acoustic similarity, because of possible varia-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 504–511,
Prague, Czech Republic, June 2007.

tions in speaking rate or pronunciation. Moreover, in the absence of higher-level knowledge, information about word boundaries is not always discernible from the raw acoustic input. This causes problems because we have no obvious unit of comparison. Finally, noise inherent in the acoustic matching procedure complicates the detection of distributional changes in the comparison matrix.

The algorithm presented in this paper demonstrates the feasibility of topic segmentation over raw acoustic input corresponding to a single speaker. We first apply a variant of the dynamic time warping algorithm to find similar fragments in the speech input through alignment. Next, we construct a comparison matrix that aggregates the output of the alignment stage. Since aligned utterances are separated by gaps and differ in duration, this representation gives rise to sparse and irregular input. To obtain robust similarity change detection, we invoke a series of transformations to smooth and refine the comparison matrix. Finally, we apply the minimum-cut segmentation algorithm to the transformed comparison matrix to detect topic boundaries.

We compare the performance of our method against traditional transcript-based segmentation algorithms. As expected, the performance of the latter depends on the accuracy of the input transcript. When a manual transcription is available, the gap between audio-based segmentation and transcript-based segmentation is substantial. However, in a more realistic scenario when the transcripts are fraught with recognition errors, the two approaches exhibit similar performance. These results demonstrate that audio-based algorithms are an effective and efficient solution for applications where transcripts are unavailable or highly errorful.

2 Related Work

Speech-based Topic Segmentation A variety of supervised and unsupervised methods have been employed to segment speech input. Some of these algorithms have been originally developed for processing written text (Beeferman et al., 1999). Others are specifically adapted for processing speech input by adding relevant acoustic features such as pause length and speaker change (Galley et al., 2003; Dielmann and Renals, 2005). In parallel, researchers extensively study the relationship between discourse structure and intonational variation (Hirschberg and Nakatani, 1996; Shriberg et al., 2000). However, all of the existing segmentation methods require as input a speech transcript of reasonable quality. In contrast, the method presented in this paper does not assume the availability of transcripts, which prevents us from using segmentation algorithms developed for written text.

At the same time, our work is closely related to unsupervised approaches for text segmentation. The central assumption here is that sharp changes in lexical distribution signal the presence of topic boundaries (Hearst, 1994; Choi et al., 2001). These approaches determine segment boundaries by identifying homogeneous regions within a similarity matrix that encodes pairwise similarity between textual units, such as sentences. Our segmentation algorithm operates over a distortion matrix, but the unit of comparison is the speech signal over a time interval. This change in representation gives rise to multiple challenges related to the inherent noise of acoustic matching, and requires the development of new methods for signal discretization, interval comparison and matrix analysis.

Pattern Induction in Acoustic Data Our work is related to research on unsupervised lexical acquisition from continuous speech. These methods aim to infer vocabulary from unsegmented audio streams by analyzing regularities in pattern distribution (de Marcken, 1996; Brent, 1999; Venkataraman, 2001). Traditionally, the speech signal is first converted into a string-like representation such as phonemes and syllables using a phonetic recognizer.

Park and Glass (2006) have recently shown the feasibility of an audio-based approach for word discovery. They induce the vocabulary from the audio stream directly, avoiding the need for phonetic transcription. Their method can accurately discover words which appear with high frequency in the audio stream. While the results obtained by Park and Glass inspire our approach, we cannot directly use their output as proxies for words in topic segmentation. Many of the content words occurring only a few times in the text are pruned away by this method. Our results show that this data that is too sparse and noisy for robustly discerning changes in

lexical distribution.

3 Algorithm

The audio-based segmentation algorithm identifies topic boundaries by analyzing changes in the distribution of acoustic patterns. The analysis is performed in three steps. First, we identify recurring patterns in the audio stream and compute distortion between them (Section 3.1). These acoustic patterns correspond to high-frequency words and phrases, but they only cover a fraction of the words that appear in the input. As a result, the distributional profile obtained during this process is too sparse to deliver robust topic analysis. Second, we generate an acoustic comparison matrix that aggregates information from multiple pattern matches (Section 3.2). Additional matrix transformations during this step reduce the noise and irregularities inherent in acoustic matching. Third, we partition the matrix to identify segments with a homogeneous distribution of acoustic patterns (Section 3.3).

3.1 Comparing Acoustic Patterns

Given a raw acoustic waveform, we extract a set of acoustic patterns that occur frequently in the speech document. Continuous speech includes many word sequences that lack clear low-level acoustic cues to denote word boundaries. Therefore, we cannot perform this task through simple counting of speech segments separated by silence. Instead, we use a local alignment algorithm to search for similar speech segments and quantify the amount of distortion between them. In what follows, we first present a vector representation used in this computation, and then specify the alignment algorithm that finds similar segments.

MFCC Representation We start by transforming the acoustic signal into a vector representation that facilitates the comparison of acoustic sequences. First, we perform silence detection on the original waveform by registering a pause if the energy falls below a certain threshold for a duration of 2s. This enables us to break up the acoustic stream into continuous spoken utterances.

This step is necessary as it eliminates spurious alignments between silent regions of the acoustic waveform. Note that silence detection is not equivalent to word boundary detection, as segmentation by silence detection alone only accounts for 20% of word boundaries in our corpus.

Next, we convert each utterance into a time series of vectors consisting of Mel-scale cepstral coefficients (MFCCs). This compact low-dimensional representation is commonly used in speech processing applications because it approximates human auditory models.

The process of extracting MFCCs from the speech signal can be summarized as follows. First, the 16 kHz digitized audio waveform is normalized by removing the mean and scaling the peak amplitude. Next, the short-time Fourier transform is taken at a frame interval of 10 ms using a 25.6 ms Hamming window. The spectral energy from the Fourier transform is then weighted by Mel-frequency filters (Huang et al., 2001). Finally, the discrete cosine transform of the log of these Mel-frequency spectral coefficients is computed, yielding a series of 14-dimensional MFCC vectors. We take the additional step of whitening the feature vectors, which normalizes the variance and decorrelates the dimensions of the feature vectors (Bishop, 1995). This whitened spectral representation enables us to use the standard unweighted Euclidean distance metric. After this transformation, the distances in each dimension will be uncorrelated and have equal variance.

Alignment Now, our goal is to identify acoustic patterns that occur multiple times in the audio waveform. The patterns may not be repeated exactly, but will most likely reoccur in varied forms. We capture this information by extracting pairs of patterns with an associated distortion score. The computation is performed using a sequence alignment algorithm.

Table 1 shows examples of alignments automatically computed by our algorithm. The corresponding phonetic transcriptions[1] demonstrate that the matching procedure can robustly handle variations in pronunciations. For example, two instances of the word *"direction"* are matched to one another despite different pronunciations, ("d ay" vs. "d ax" in the first syllable). At the same time, some aligned pairs form erroneous matches, such as *"my prediction"* matching *"y direction"* due to their high acoustic

[1] Phonetic transcriptions are not used by our algorithm and are provided for illustrative purposes only.

Aligned Word(s)	Phonetic Transcription
the x direction	dh iy eh kcl k s dcl d ax r eh kcl sh ax n
	ð iʸ ɛkˀk s dˀd ər ɛkˀʃən
the y direction	dh ax w ay dcl d ay r eh kcl sh epi en
	ð əw aʸ dˀaʸ r ɛkˀk ʃən
of my prediction	ax v m ay kcl k r iy l iy kcl k sh ax n
	əv m aʸ kˀk r iʸ l iʸ kˀk ʃən
acceleration	eh kcl k s eh l ax r ey sh epi en
	ɛkˀk s ɛl ər εʸ ʃ- n̩
acceleration	ax kcl k s ah n ax r eh n epi sh epi en
	əkˀk s ʌn ər ɛn - ʃ- n̩
the derivation	dcl d ih dx ih z dcl dh ey sh epi en
	dˀd ɪrɪz dˀð εʸ ʃ- n̩
a demonstration	uh dcl d eh m ax n epi s tcl t r ey sh en
	ʊdˀd ɛm ən - s tˀt r εʸ ʃn̩

Table 1: Aligned Word Paths. Each group of rows represents audio segments that were aligned to one another, along with their corresponding phonetic transcriptions using TIMIT conventions (Garofolo et al., 1993) and their IPA equivalents.

similarity.

The alignment algorithm operates on the audio waveform represented by a list of silence-free utterances $(u_1, u_2, \ldots, u_n)$. Each utterance u' is a time series of MFCC vectors $(\vec{x'_1}, \vec{x'_2}, \ldots, \vec{x'_m})$. Given two input utterances u' and u'', the algorithm outputs a set of alignments between the corresponding MFCC vectors. The alignment distortion score is computed by summing the Euclidean distances of matching vectors.

To compute the optimal alignment we use a variant of the dynamic time warping algorithm (Huang et al., 2001). For every possible starting alignment point, we optimize the following dynamic programming objective:

$$D(i_k, j_k) = d(i_k, j_k) + \min \begin{cases} D(i_k - 1, j_k) \\ D(i_k, j_k - 1) \\ D(i_k - 1, j_k - 1) \end{cases}$$

In the equation above, i_k and j_k are alignment endpoints in the k-th subproblem of dynamic programming.

This objective corresponds to a descent through a dynamic programming trellis by choosing right, down, or diagonal steps at each stage.

During the search process, we consider not only the alignment distortion score, but also the shape of the alignment path. To limit the amount of temporal warping, we enforce the following constraint:

$$\left|(i_k - i_1) - (j_k - j_1)\right| \leq R, \forall k, \qquad (1)$$

$$i_k \leq N_x \quad \text{and} \quad j_k \leq N_y,$$

where N_x and N_y are the number of MFCC samples in each utterance. The value $2R + 1$ is the width of the diagonal band that controls the extent of temporal warping. The parameter R is tuned on a development set.

This alignment procedure may produce paths with high distortion subpaths. Therefore, we trim each path to retain the subpath with lowest average distortion and length at least L. More formally, given an alignment of length N, we seek to find m and n such that:

$$\underset{1 \leq m \leq n \leq N}{\arg\min} \frac{1}{n - m + 1} \sum_{k=m}^{n} d(i_k, j_k) \qquad n - m \geq L$$

We accomplish this by computing the length constrained minimum average distortion subsequence of the path sequence using an $O(N \log(L))$ algorithm proposed by Lin et al (2002). The length parameter, L, allows us to avoid overtrimming and control the length of alignments that are found. After trimming, the distortion of each alignment path is normalized by the path length.

Alignments with a distortion exceeding a prespecified threshold are pruned away to ensure that the aligned phrasal units are close acoustic matches. This parameter is tuned on a development set.

In the next section, we describe how to aggregate information from multiple noisy matches into a representation that facilitates boundary detection.

3.2 Construction of Acoustic Comparison Matrix

The goal of this step is to construct an acoustic comparison matrix that will guide topic segmentation. This matrix encodes variations in the distribution of acoustic patterns for a given speech document. We construct this matrix by first discretizing the acoustic signal into constant-length blocks and then computing the distortion between pairs of blocks.

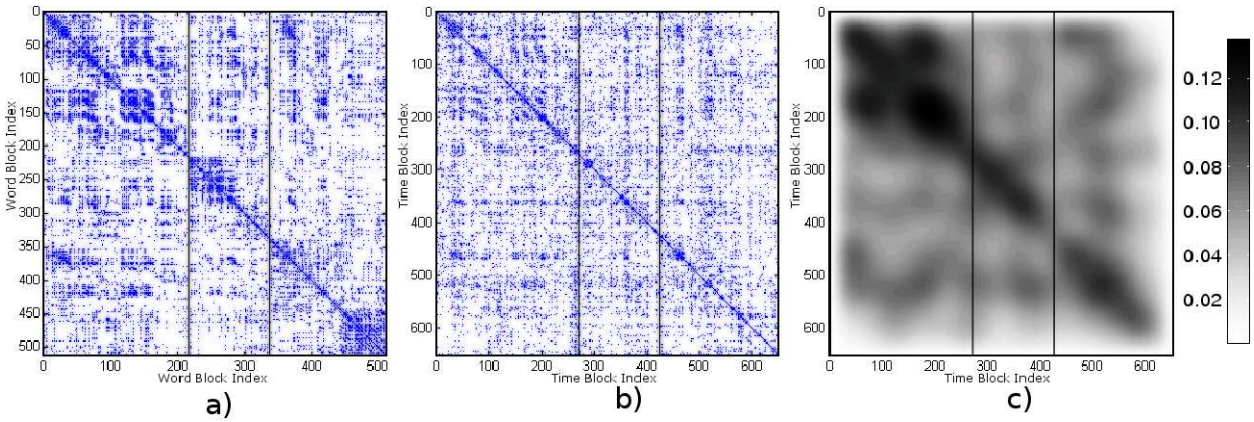


Figure 1: a) Similarity matrix for a Physics lecture constructed using a manual transcript. b) Similarity matrix for the same lecture constructed from acoustic data. The intensity of a pixel indicates the degree of block similarity. c) Acoustic comparison matrix after 2000 iterations of anisotropic diffusion. Vertical lines correspond to the reference segmentation.

Unfortunately, the paths and distortions generated during the alignment step (Section 3.1) cannot be mapped directly to an acoustic comparison matrix. Since we compare only commonly repeated acoustic patterns, some portions of the signal correspond to gaps between alignment paths. In fact, in our corpus only 67% of the data is covered by alignment paths found during the alignment stage. Moreover, many of these paths are not disjoint. For instance, our experiments show that 74% of them overlap with at least one additional alignment path. Finally, these alignments vary significantly in duration, ranging from 0.350 ms to 2.7 ms in our corpus.

Discretization and Distortion Computation To compensate for the irregular distribution of alignment paths, we quantize the data by splitting the input signal into uniform contiguous time blocks. A time block does not necessarily correspond to any one discovered alignment path. It may contain several complete paths and also portions of other paths. We compute the aggregate distortion score $D(x, y)$ of two blocks x and y by summing the distortions of all alignment paths that fall within x and y.

Matrix Smoothing Equipped with a block distortion measure, we can now construct an acoustic comparison matrix. In principle, this matrix can be processed employing standard methods developed for text segmentation. However, as Figure 1 illustrates, the structure of the acoustic matrix is quite different from the one obtained from text. In a transcript similarity matrix shown in Figure 1 a), reference boundaries delimit homogeneous regions with high internal similarity. On the other hand, looking at the acoustic similarity matrix[2] shown in Figure 1 b), it is difficult to observe any block structure corresponding to the reference segmentation.

This deficiency can be attributed to the sparsity of acoustic alignments. Consider, for example, the case when a segment is interspersed with blocks that contain very few or no complete paths. Even though the rest of the blocks in the segment could be closely related, these path-free blocks dilute segment homogeneity. This is problematic because it is not always possible to tell whether a sudden shift in scores signifies a transition or if it is just an artifact of irregularities in acoustic matching. Without additional matrix processing, these irregularities will lead the system astray.

We further refine the acoustic comparison matrix using *anisotropic diffusion*. This technique has been developed for enhancing edge detection accuracy in image processing (Perona and Malik, 1990), and has been shown to be an effective smoothing method in text segmentation (Ji and Zha, 2003). When applied to a comparison matrix, anisotropic diffusion reduces score variability within homogeneous re-

[2]We converted the original comparison distortion matrix to the similarity matrix by subtracting the component distortions from the maximum alignment distortion score.

gions of the matrix and makes edges between these regions more pronounced. Consequently, this transformation facilitates boundary detection, potentially increasing segmentation accuracy. In Figure 1 c), we can observe that the boundary structure in the diffused comparison matrix becomes more salient and corresponds more closely to the reference segmentation.

3.3 Matrix Partitioning

Given a target number of segments k, the goal of the partitioning step is to divide a matrix into k square submatrices along the diagonal. This process is guided by an optimization function that maximizes the homogeneity within a segment or minimizes the homogeneity across segments. This optimization problem can be solved using one of many unsupervised segmentation approaches (Choi et al., 2001; Ji and Zha, 2003; Malioutov and Barzilay, 2006).

In our implementation, we employ the minimum-cut segmentation algorithm (Shi and Malik, 2000; Malioutov and Barzilay, 2006). In this graph-theoretic framework, segmentation is cast as a problem of partitioning a weighted undirected graph that minimizes the *normalized-cut criterion*. The minimum-cut method achieves robust analysis by jointly considering all possible partitionings of a document, moving beyond localized decisions. This allows us to aggregate comparisons from multiple locations, thereby compensating for the noise of individual matches.

4 Evaluation Set-Up

Data We use a publicly available[3] corpus of introductory Physics lectures described in our previous work (Malioutov and Barzilay, 2006). This material is a particularly appealing application area for an audio-based segmentation algorithm — many academic subjects lack transcribed data for training, while a high ratio of in-domain technical terms limits the use of out-of-domain transcripts. This corpus is also challenging from the segmentation perspective because the lectures are long and transitions between topics are subtle.

[3]See `http://www.csail.mit.edu/~igorm/acl06.html`

The corpus consists of 33 lectures, with an average length of 8500 words and an average duration of 50 minutes. On average, a lecture was annotated with six segments, and a typical segment corresponds to two pages of a transcript. Three lectures from this set were used for development, and 30 lectures were used for testing. The lectures were delivered by the same speaker.

To evaluate the performance of traditional transcript-based segmentation algorithms on this corpus, we also use several types of transcripts at different levels of recognition accuracy. In addition to manual transcripts, our corpus contains two types of automatic transcripts, one obtained using speaker-dependent (SD) models and the other obtained using speaker-independent (SI) models. The speaker-independent model was trained on 85 hours of out-of-domain general lecture material and contained no speech from the speaker in the test set. The speaker-dependent model was trained by using 38 hours of audio data from other lectures given by the speaker. Both recognizers incorporated word statistics from the accompanying class textbook into the language model. The word error rates for the speaker-independent and speaker-dependent models are 44.9% and 19.4%, respectively.

Evaluation Metrics We use the P_k and WindowDiff measures to evaluate our system (Beeferman et al., 1999; Pevzner and Hearst, 2002). The P_k measure estimates the probability that a randomly chosen pair of words within a window of length k words is inconsistently classified. The WindowDiff metric is a variant of the P_k measure, which penalizes false positives and near misses equally. For both of these metrics, lower scores indicate better segmentation accuracy.

Baseline We use the state-of-the-art mincut segmentation system by Malioutov and Barzilay (2006) as our point of comparison. This model is an appropriate baseline, because it has been shown to compare favorably with other top-performing segmentation systems (Choi et al., 2001; Utiyama and Isahara, 2001). We use the publicly available implementation of the system.

As additional points of comparison, we test the uniform and random baselines. These correspond to segmentations obtained by uniformly placing

	P_k	WindowDiff
MAN	0.298	0.311
SD	0.340	0.351
AUDIO	**0.358**	**0.370**
SI	0.378	0.390
RAND	0.472	0.497
UNI	0.476	0.484

Table 2: Segmentation accuracy for audio-based segmentor (AUDIO), random (RAND), uniform (UNI) and three transcript-based segmentation algorithms that use manual (MAN), speaker-dependent (SD) and speaker-independent (SI) transcripts. For all of the algorithms, the target number of segments is set to the reference number of segments.

boundaries along the span of the lecture and selecting random boundaries, respectively.

To control for segmentation granularity, we specify the number of segments in the reference segmentation for both our system and the baselines.

Parameter Tuning We tuned the number of quantized blocks, the edge cutoff parameter of the minimum cut algorithm, and the anisotropic diffusion parameters on a heldout set of three development lectures. We used the same development set for the baseline segmentation systems.

5 Results

The goal of our evaluation experiments is two-fold. First, we are interested in understanding the conditions in which an audio-based segmentation is advantageous over a transcript-based one. Second, we aim to analyze the impact of various design decisions on the performance of our algorithm.

Comparison with Transcript-Based Segmentation Table 2 shows the segmentation accuracy of the audio-based segmentation algorithm and three transcript-based segmentors on the set of 30 Physics lectures. Our algorithm yields an average P_k measure of 0.358 and an average WindowDiff measure of 0.370. This result is markedly better than the scores attained by uniform and random segmentations. As expected, the best segmentation results are obtained using manual transcripts. However, the gap between audio-based segmentation and transcript-based segmentation narrows when the recognition accuracy decreases. In fact, performance of the audio-based segmentation beats the transcript-based segmentation baseline obtained using speaker-independent (SI) models (0.358 for AUDIO versus P_k measurements of 0.378 for SI).

Analysis of Audio-based Segmentation A central challenge in audio-based segmentation is how to overcome the noise inherent in acoustic matching. We addressed this issue by using anisotropic diffusion to refine the comparison matrix. We can quantify the effects of this smoothing technique by generating segmentations directly from the similarity matrix. We obtain similarities from the distortions in the comparison matrix by subtracting the distortion scores from the maximum distortion:

$$S(x, y) = \max_{s_i, s_j} [D(s_i, s_j)] - D(x, y)$$

Using this matrix with the min-cut algorithm, segmentation accuracy drops to a P_k measure of 0.418 (0.450 WindowDiff). This difference in performance shows that anisotropic diffusion compensates for noise introduced during acoustic matching.

An alternative solution to the problem of irregularities in audio-based matching is to compute clusters of acoustically similar utterances. Each of the derived clusters can be thought of as a unique word type.[4] We compute these clusters, employing a method for unsupervised vocabulary induction developed by Park and Glass (2006). Using the output of their algorithm, the continuous audio stream is transformed into a sequence of word-like units, which in turn can be segmented using any standard transcript-based segmentation algorithm, such as the minimum-cut segmentor. On our corpus, this method achieves disappointing results — a P_k measure of 0.423 (0.424 WindowDiff). The result can be attributed to the sparsity of clusters[5] generated by this method, which focuses primarily on discovering the frequently occurring content words.

6 Conclusion and Future Work

We presented an unsupervised algorithm for audio-based topic segmentation. In contrast to existing

[4] In practice, a cluster can correspond to a phrase, word, or word fragment (See Table 1 for examples).

[5] We tuned the number of clusters on the development set.

algorithms for speech segmentation, our approach does not require an input transcript. Thus, it can be used in domains where a speech recognizer is not available or its output is too noisy. Our approach approximates the distribution of cohesion ties by considering the distribution of acoustic patterns. Our experimental results demonstrate the utility of this approach: audio-based segmentation compares favorably with transcript-based segmentation computed over noisy transcripts.

The segmentation algorithm presented in this paper focuses on one source of linguistic information for discourse analysis — lexical cohesion. Multiple studies of discourse structure, however, have shown that prosodic cues are highly predictive of changes in topic structure (Hirschberg and Nakatani, 1996; Shriberg et al., 2000). In a supervised framework, we can further enhance audio-based segmentation by combining features derived from pattern analysis with prosodic information. We can also explore an unsupervised fusion of these two sources of information; for instance, we can induce informative prosodic cues by using distributional evidence.

Another interesting direction for future research lies in combining the results of noisy recognition with information obtained from distribution of acoustic patterns. We hypothesize that these two sources provide complementary information about the audio stream, and therefore can compensate for each other's mistakes. This combination can be particularly fruitful when processing speech documents with multiple speakers or background noise.

7 Acknowledgements

The authors acknowledge the support of the Microsoft Faculty Fellowship and the National Science Foundation (CAREER grant IIS-0448168, grant IIS-0415865, and the NSF Graduate Fellowship). Any opinions, findings, conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. We would like to thank T.J. Hazen for his assistance with the speech recognizer and to acknowledge Tara Sainath, Natasha Singh, Ben Snyder, Chao Wang, Luke Zettlemoyer and the three anonymous reviewers for their valuable comments and suggestions.

References

D. Beeferman, A. Berger, J. D. Lafferty. 1999. Statistical models for text segmentation. *Machine Learning*, 34(1-3):177–210.

C. Bishop, 1995. *Neural Networks for Pattern Recognition*, pg. 38. Oxford University Press, New York, 1995.

M. R. Brent. 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34(1-3):71–105.

F. Choi, P. Wiemer-Hastings, J. Moore. 2001. Latent semantic analysis for text segmentation. In *Proceedings of EMNLP*, 109–117.

C. G. de Marcken. 1996. *Unsupervised Language Acquisition*. Ph.D. thesis, Massachusetts Institute of Technology.

A. Dielmann, S. Renals. 2005. Multistream dynamic Bayesian network for meeting segmentation. In *Proceedings Multimodal Interaction and Related Machine Learning Algorithms Workshop (MLMI–04)*, 76–86.

M. Galley, K. McKeown, E. Fosler-Lussier, H. Jing. 2003. Discourse segmentation of multi-party conversation. In *Proceedings of the ACL*, 562–569.

J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallet, N. Dahlgren, V. Zue. 1993. TIMIT Acoustic-Phonetic Continuous Speech Corpus. Linguistic Data Consortium, 1993.

M. Hearst. 1994. Multi-paragraph segmentation of expository text. In *Proceedings of the ACL*, 9–16.

J. Hirschberg, C. H. Nakatani. 1996. A prosodic analysis of discourse segments in direction-giving monologues. In *Proceedings of the ACL*, 286–293.

X. Huang, A. Acero, H.-W. Hon. 2001. *Spoken Language Processing*. Prentice Hall.

X. Ji, H. Zha. 2003. Domain-independent text segmentation using anisotropic diffusion and dynamic programming. In *Proceedings of SIGIR*, 322–329.

Y.-L. Lin, T. Jiang, K.-M. Chao. 2002. Efficient algorithms for locating the length-constrained heaviest segments with applications to biomolecular sequence analysis. *J. Computer and System Sciences*, 65(3):570–586.

I. Malioutov, R. Barzilay. 2006. Minimum cut model for spoken lecture segmentation. In *Proceedings of the COLING/ACL*, 25–32.

A. Park, J. R. Glass. 2006. Unsupervised word acquisition from speech using pattern discovery. In *Proceedings of ICASSP*.

P. Perona, J. Malik. 1990. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639.

L. Pevzner, M. Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36.

J. Shi, J. Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905.

E. Shriberg, A. Stolcke, D. Hakkani-Tur, G. Tur. 2000. Prosody-based automatic segmentation of speech into sentences and topics. *Speech Communication*, 32(1-2):127–154.

M. Utiyama, H. Isahara. 2001. A statistical model for domain-independent text segmentation. In *Proceedings of the ACL*, 499–506.

A. Venkataraman. 2001. A statistical model for word discovery in transcribed speech. *Computational Linguistics*, 27(3):353–372.

Randomised Language Modelling for Statistical Machine Translation

David Talbot and Miles Osborne
School of Informatics, University of Edinburgh
2 Buccleuch Place, Edinburgh, EH8 9LW, UK
d.r.talbot@sms.ed.ac.uk, miles@inf.ed.ac.uk

Abstract

A Bloom filter (BF) is a randomised data structure for set membership queries. Its space requirements are significantly below lossless information-theoretic lower bounds but it produces false positives with some quantifiable probability. Here we explore the use of BFs for language modelling in statistical machine translation.

We show how a BF containing n-grams can enable us to use much larger corpora and higher-order models complementing a conventional n-gram LM within an SMT system. We also consider (i) how to include approximate frequency information efficiently within a BF and (ii) how to reduce the error rate of these models by first checking for lower-order sub-sequences in candidate n-grams. Our solutions in both cases retain the one-sided error guarantees of the BF while taking advantage of the Zipf-like distribution of word frequencies to reduce the space requirements.

1 Introduction

Language modelling (LM) is a crucial component in statistical machine translation (SMT). Standard n-gram language models assign probabilities to translation hypotheses in the target language, typically as smoothed trigram models, e.g. (Chiang, 2005). Although it is well-known that higher-order LMs and models trained on additional monolingual corpora can yield better translation performance, the challenges in deploying large LMs are not trivial. Increasing the order of an n-gram model can result in an exponential increase in the number of parameters; for corpora such as the English Gigaword corpus, for instance, there are 300 million distinct trigrams and over 1.2 billion 5-grams. Since a LM may be queried millions of times per sentence, it should ideally reside locally in memory to avoid time-consuming remote or disk-based look-ups.

Against this background, we consider a radically different approach to language modelling: instead of explicitly storing all distinct n-grams, we store a randomised representation. In particular, we show that the *Bloom filter* (Bloom (1970); BF), a simple space-efficient randomised data structure for representing sets, may be used to represent statistics from larger corpora and for higher-order n-grams to complement a conventional smoothed trigram model within an SMT decoder. [1]

The space requirements of a Bloom filter are quite spectacular, falling significantly below information-theoretic error-free lower bounds while query times are constant. This efficiency, however, comes at the price of *false positives*: the filter may erroneously report that an item not in the set is a member. False negatives, on the other hand, will never occur: the error is said to be *one-sided*.

In this paper, we show that a Bloom filter can be used effectively for language modelling within an SMT decoder and present the *log-frequency Bloom filter*, an extension of the standard Boolean BF that

[1] For extensions of the framework presented here to stand-alone smoothed Bloom filter language models, we refer the reader to a companion paper (Talbot and Osborne, 2007).

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 512–519,
Prague, Czech Republic, June 2007.

takes advantage of the Zipf-like distribution of corpus statistics to allow frequency information to be associated with n-grams in the filter in a space-efficient manner. We then propose a mechanism, *sub-sequence filtering*, for reducing the error rates of these models by using the fact that an n-gram's frequency is bound from above by the frequency of its least frequent sub-sequence.

We present machine translation experiments using these models to represent information regarding higher-order n-grams and additional larger monolingual corpora in combination with conventional smoothed trigram models. We also run experiments with these models in isolation to highlight the impact of different order n-grams on the translation process. Finally we provide some empirical analysis of the effectiveness of both the log frequency Bloom filter and sub-sequence filtering.

2 The Bloom filter

In this section, we give a brief overview of the Bloom filter (BF); refer to Broder and Mitzenmacher (2005) for a more in detailed presentation. A BF represents a set $\mathcal{S} = \{x_1, x_2, ..., x_n\}$ with n elements drawn from a universe $\mathcal{U}$ of size N. The structure is attractive when $N \gg n$. The only significant storage used by a BF consists of a bit array of size m. This is initially set to hold zeroes. To train the filter we hash each item in the set k times using distinct hash functions $h_1, h_2, ..., h_k$. Each function is assumed to be independent from each other and to map items in the universe to the range 1 to m uniformly at random. The k bits indexed by the hash values for each item are set to 1; the item is then discarded. Once a bit has been set to 1 it remains set for the lifetime of the filter. Distinct items may not be hashed to k distinct locations in the filter; we ignore collisons. Bits in the filter can, therefore, be *shared* by distinct items allowing significant space savings but introducing a non-zero probability of false positives at test time. There is no way of directly retrieving or ennumerating the items stored in a BF.

At test time we wish to discover whether a given item was a member of the original set. The filter is queried by hashing the test item using the same k hash functions. If all bits referenced by the k hash values are 1 then we *assume* that the item was a member; if any of them are 0 then we *know* it was not. True members are always correctly identified, but a false positive will occur if all k corresponding bits were set by other items during training and the item was not a member of the training set. This is known as a *one-sided error*.

The probability of a false postive, f, is clearly the probability that none of k randomly selected bits in the filter are still 0 after training. Letting p be the proportion of bits that are still zero after these n elements have been inserted, this gives,

$$f = (1-p)^k.$$

As n items have been entered in the filter by hashing each k times, the probability that a bit is still zero is,

$$p^{'} = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{-\frac{kn}{m}}$$

which is the expected value of p. Hence the false positive rate can be approximated as,

$$f = (1-p)^k \approx (1-p^{'})^k \approx \left(1 - e^{-\frac{kn}{m}}\right)^k.$$

By taking the derivative we find that the number of functions k^* that minimizes f is,

$$k^* = \ln 2 \cdot \frac{m}{n}.$$

which leads to the intuitive result that exactly half the bits in the filter will be set to 1 when the optimal number of hash functions is chosen.

The fundmental difference between a Bloom filter's space requirements and that of any lossless representation of a set is that the former does not depend on the size of the (exponential) universe N from which the set is drawn. A lossless representation scheme (for example, a hash map, trie etc.) must depend on N since it assigns a distinct representation to each possible set drawn from the universe.

3 Language modelling with Bloom filters

In our experiments we make use of both standard (i.e. Boolean) BFs containing n-gram types drawn from a training corpus and a novel BF scheme, the *log-frequency Bloom filter*, that allows frequency information to be associated efficiently with items stored in the filter.

Algorithm 1 Training frequency BF

Input: $\mathcal{S}_{train}$, $\{h_1, ...h_k\}$ and $\mathcal{BF} = \emptyset$
Output: $\mathcal{BF}$
for all $x \in \mathcal{S}_{train}$ **do**
  $c(x) \leftarrow$ frequency of n-gram x in $\mathcal{S}_{train}$
  $qc(x) \leftarrow$ quantisation of $c(x)$ (Eq. 1)
  for $j = 1$ to $qc(x)$ **do**
    for $i = 1$ to k **do**
      $h_i(x) \leftarrow$ hash of event $\{x, j\}$ under h_i
      $\mathcal{BF}[h_i(x)] \leftarrow 1$
    end for
  end for
end for
return $\mathcal{BF}$

Algorithm 2 Test frequency BF

Input: x, $MAXQCOUNT$, $\{h_1, ...h_k\}$ and $\mathcal{BF}$
Output: Upper bound on $qc(x) \in \mathcal{S}_{train}$
for $j = 1$ to $MAXQCOUNT$ **do**
  for $i = 1$ to k **do**
    $h_i(x) \leftarrow$ hash of event $\{x, j\}$ under h_i
    if $\mathcal{BF}[h_i(x)] = 0$ **then**
      return $j - 1$
    end if
  end for
end for

3.1 Log-frequency Bloom filter

The efficiency of our scheme for storing n-gram statistics within a BF relies on the Zipf-like distribution of n-gram frequencies in natural language corpora: most events occur an extremely small number of times, while a small number are very frequent.

We quantise raw frequencies, $c(x)$, using a logarithmic codebook as follows,

$$qc(x) = 1 + \lfloor \log_b c(x) \rfloor. \quad (1)$$

The precision of this codebook decays exponentially with the raw counts and the scale is determined by the base of the logarithm b; we examine the effect of this parameter in experiments below.

Given the quantised count $qc(x)$ for an n-gram x, the filter is trained by entering composite events consisting of the n-gram appended by an integer counter j that is incremented from 1 to $qc(x)$ into the filter. To retrieve the quantised count for an n-gram, it is first appended with a count of 1 and hashed under the k functions; if this tests positive, the count is incremented and the process repeated. The procedure terminates as soon as any of the k hash functions hits a 0 and the previous count is reported. The one-sided error of the BF and the training scheme ensure that the actual quantised count cannot be larger than this value. As the counts are quantised logarithmically, the counter will be incremented only a small number of times. The training and testing routines are given here as Algorithms 1 and 2 respectively.

Errors for the log-frequency BF scheme are one-sided: frequencies will never be underestimated. The probability of overestimating an item's frequency decays exponentially with the size of the overestimation error d (i.e. as f^d for $d > 0$) since each erroneous increment corresponds to a single false positive and d such independent events must occur together.

3.2 Sub-sequence filtering

The error analysis in Section 2 focused on the false positive rate of a BF; if we deploy a BF within an SMT decoder, however, the actual error rate will also depend on the *a priori* membership probability of items presented to it. The error rate Err is,

$$Err = Pr(x \notin S_{train}|Decoder)f.$$

This implies that, unlike a conventional lossless data structure, the model's accuracy depends on other components in system and how it is queried.

We take advantage of the monotonicity of the n-gram event space to place upper bounds on the frequency of an n-gram prior to testing for it in the filter and potentially truncate the outer loop in Algorithm 2 when we know that the test could only return postive in error.

Specifically, if we have stored lower-order n-grams in the filter, we can infer that an n-gram cannot present, if any of its sub-sequences test negative. Since our scheme for storing frequencies can never *underestimate* an item's frequency, this relation will generalise to frequencies: an n-gram's frequency cannot be greater than the frequency of its least frequent sub-sequence as reported by the filter,

$$c(w_1, ..., w_n) \leq \min\{c(w_1, ..., w_{n-1}), c(w_2, ..., w_n)\}.$$

We use this to reduce the effective error rate of BF-LMs that we use in the experiments below.

3.3 Bloom filter language model tests

A standard BF can implement a Boolean 'language model' test: have we seen some fragment of language before? This does not use any frequency information. The *Boolean BF-LM* is a standard BF containing all n-grams of a certain length in the training corpus, $\mathcal{S}_{train}$. It implements the following binary feature function in a log-linear decoder,

$$\phi_{\mathbf{bool}}(x) \geq \delta(x \in \mathcal{S}_{train})$$

Separate Boolean BF-LMs can be included for different order n and assigned distinct log-linear weights that are learned as part of a minimum error rate training procedure (see Section 4).

The *log-frequency BF-LM* implements a multinomial feature function in the decoder that returns the value associated with an n-gram by Algorithm 2.

$$\phi_{\mathbf{logfreq}}(x) \geq qc(x) \in \mathcal{S}_{train}$$

Sub-sequence filtering can be performed by using the minimum value returned by lower-order models as an upper-bound on the higher-order models.

By boosting the score of hypotheses containing n-grams observed in the training corpus while remaining agnostic for unseen n-grams (with the exception of errors), these feature functions have more in common with maximum entropy models than conventionally smoothed n-gram models.

4 Experiments

We conducted a range of experiments to explore the effectiveness and the error-space trade-off of Bloom filters for language modelling in SMT. The space-efficiency of these models also allows us to investigate the impact of using much larger corpora and higher-order n-grams on translation quality. While our main experiments use the Bloom filter models *in conjunction* with a conventional smoothed trigram model, we also present experiments with these models in isolation to highlight the impact of different order n-grams on the translation process. Finally, we present some empirical analysis of both the log-frequency Bloom filter and the sub-sequence filtering technique which may be of independent interest.

Model	EP-KN-3	EP-KN-4	AFP-KN-3
Memory	64M	99M	1.3G
gzip size	21M	31M	481M
1-gms	62K	62K	871K
2-gms	1.3M	1.3M	16M
3-gms	1.1M	1.0M	31M
4-gms	N/A	1.1M	N/A

Table 1: Baseline and Comparison Models

4.1 Experimental set-up

All of our experiments use publically available resources. We use the French-English section of the *Europarl* (EP) corpus for parallel data and language modelling (Koehn, 2003) and the *English Gigaword Corpus* (LDC2003T05; GW) for additional language modelling.

Decoding is carried-out using the Moses decoder (Koehn and Hoang, 2007). We hold out 500 test sentences and 250 development sentences from the parallel text for evaluation purposes. The feature functions in our models are optimised using minimum error rate training and evaluation is performed using the BLEU score.

4.2 Baseline and comparison models

Our baseline LM and other comparison models are conventional n-gram models smoothed using modified Kneser-Ney and built using the SRILM Toolkit (Stolcke, 2002); as is standard practice these models drop entries for n-grams of size 3 and above when the corresponding discounted count is less than 1. The baseline language model, EP-KN-3, is a trigram model trained on the English portion of the parallel corpus. For additional comparisons we also trained a smoothed 4-gram model on this Europarl data (EP-KN-4) and a trigram model on the Agence France Press section of the Gigaword Corpus (AFP-KN-3).

Table 1 shows the amount of memory these models take up on disk and compressed using the *gzip* utility in parentheses as well as the number of distinct n-grams of each order. We give the gzip compressed size as an optimistic lower bound on the size of any lossless representation of each model.[2]

[2]Note, in particular, that gzip compressed files do *not* support direct random access as required by our application.

Corpus	Europarl	Gigaword
1-gms	61K	281K
2-gms	1.3M	5.4M
3-gms	4.7M	275M
4-gms	9.0M	599M
5-gms	10.3M	842M
6-gms	10.7M	957M

Table 2: Number of distinct n-grams

4.3 Bloom filter-based models

To create Bloom filter LMs we gathered n-gram counts from both the Europarl (EP) and the whole of the Gigaword Corpus (GW). Table 2 shows the numbers of distinct n-grams in these corpora. Note that we use no pruning for these models and that the numbers of distinct n-grams is of the same order as that of the recently released Google Ngrams dataset (LDC2006T13). In our experiments we create a range of models referred to by the corpus used (EP or GW), the order of the n-gram(s) entered into the filter (1 to 10), whether the model is Boolean (Bool-BF) or provides frequency information (Freq-BF), whether or not sub-sequence filtering was used (FTR) and whether it was used in conjunction with the baseline trigram (+EP-KN-3).

4.4 Machine translation experiments

Our first set of experiments examines the relationship between memory allocated to the BF and BLEU score. We present results using the Boolean BF-LM in isolation and then both the Boolean and log-frequency BF-LMS to add 4-grams to our baseline 3-gram model.Our second set of experiments adds 3-grams and 5-grams from the Gigaword Corpus to our baseline. Here we constrast the Boolean BF-LM with the log-frequency BF-LM with different quantisation bases (2 = fine-grained and 5 = coarse-grained). We then evaluate the sub-sequence filtering approach to reducing the actual error rate of these models by adding both 3 and 4-grams from the Gigaword Corpus to the baseline. Since the BF-LMs easily allow us to deploy very high-order n-gram models, we use them to evaluate the impact of different order n-grams on the translation process presenting results using the Boolean and log-frequency BF-LM *in isolation* for n-grams of order 1 to 10.

Model	EP-KN-3	EP-KN-4	AFP-KN-3
BLEU	28.51	29.24	29.17
Memory	64M	99M	1.3G
gzip size	21M	31M	481M

Table 3: Baseline and Comparison Models

4.5 Analysis of BF extensions

We analyse our log-frequency BF scheme in terms of the additional memory it requires and the error rate compared to a non-redundant scheme. The non-redundant scheme involves entering *just* the exact quantised count for each n-gram and then searching over the range of possible counts at test time starting with the count with maximum *a priori* probability (i.e. 1) and incrementing until a count is found or the whole codebook has been searched (here the size is 16).

We also analyse the sub-sequence filtering scheme directly by creating a BF with only 3-grams and a BF containing both 2-grams and 3-grams and comparing their *actual* error rates when presented with 3-grams that are all known to be negatives.

5 Results

5.1 Machine translation experiments

Table 3 shows the results of the baseline (EP-KN-3) and other conventional n-gram models trained on larger corpora (AFP-KN-3) and using higher-order dependencies (EP-KN-4). The larger models improve somewhat on the baseline performance.

Figure 1 shows the relationship between space allocated to the BF models and BLEU score (left) and false positive rate (right) respectively. These experiments do *not* include the baseline model. We can see a clear correlation between memory / false positive rate and translation performance.

Adding 4-grams in the form of a Boolean BF or a log-frequency BF (see Figure 2) improves on the 3-gram baseline with little additional memory (around 4MBs) while performing on a par with or above the Europarl 4-gram model with around 10MBs; this suggests that a lossy representation of the unpruned set of 4-grams contains more useful information than a lossless representation of the pruned set.[3]

[3] An unpruned modified Kneser-Ney 4-gram model on the Eurpoparl data scores slightly higher - 29.69 - while taking up 489MB (132MB gzipped).

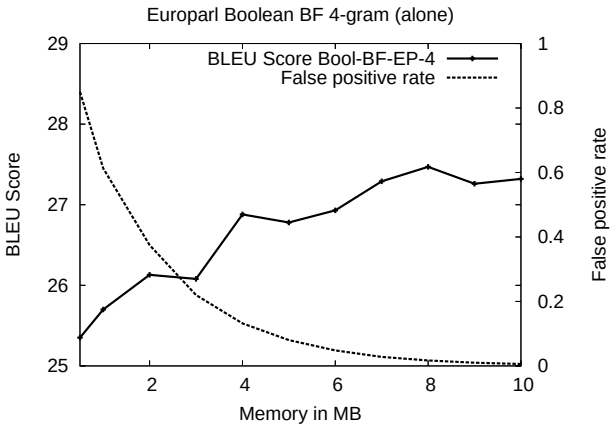


Figure 1: Space/Error vs. BLEU Score.

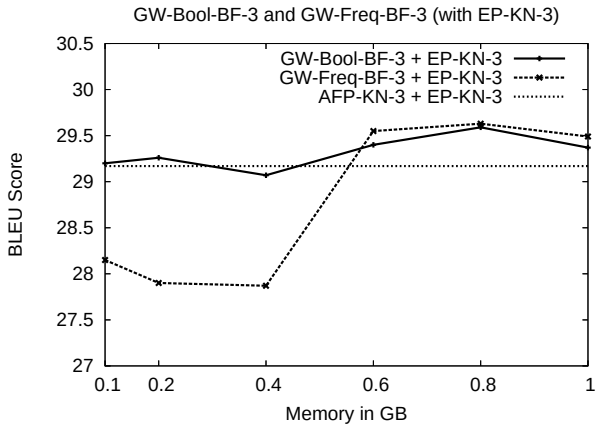


Figure 3: Adding GW 3-grams with Bloom filters.

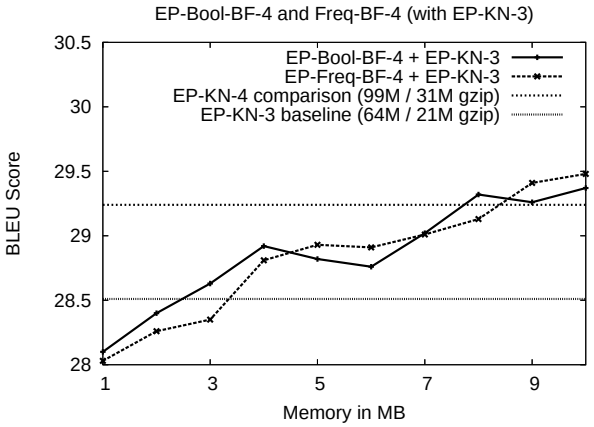


Figure 2: Adding 4-grams with Bloom filters.

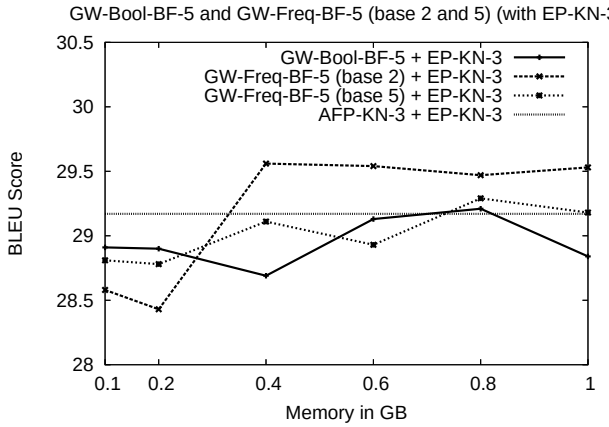


Figure 4: Comparison of different quantisation rates.

As the false positive rate exceeds 0.20 the performance is severly degraded. Adding 3-grams drawn from the whole of the Gigaword corpus rather than simply the Agence France Press section results in slightly improved performance with signficantly less memory than the AFP-KN-3 model (see Figure 3).

Figure 4 shows the results of adding 5-grams drawn from the Gigaword corpus to the baseline. It also contrasts the Boolean BF and the log-frequency BF suggesting in this case that the log-frequency BF can provide useful information when the quantisation base is relatively fine-grained (base 2). The Boolean BF and the base 5 (coarse-grained quantisation) log-frequency BF perform approximately the same. The base 2 quantisation performs worse for smaller amounts of memory, possibly due to the larger set of events it is required to store.

Figure 5 shows sub-sequence filtering resulting in a small increase in performance when false positive rates are high (i.e. less memory is allocated). We believe this to be the result of an increased *a priori* membership probability for n-grams presented to the filter under the sub-sequence filtering scheme.

Figure 6 shows that for this task the most useful n-gram sizes are between 3 and 6.

5.2 Analysis of BF extensions

Figure 8 compares the memory requirements of the log-frequencey BF (base 2) and the Boolean

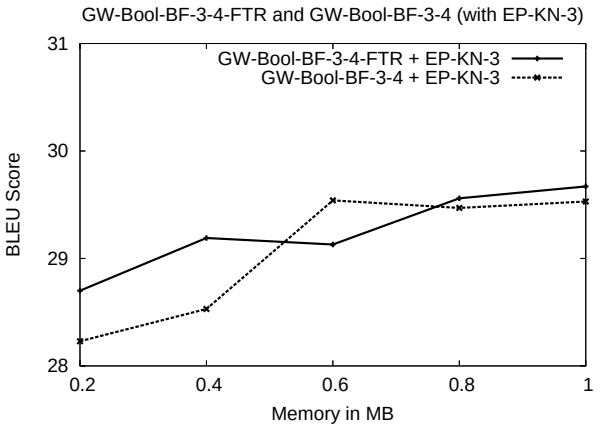


Figure 5: Effect of sub-sequence filtering.

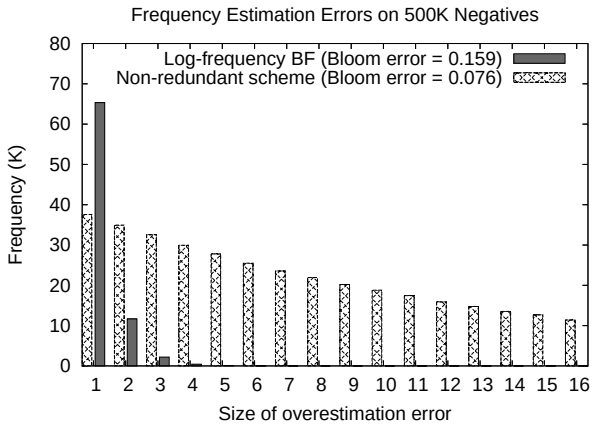


Figure 7: Frequency estimation errors.

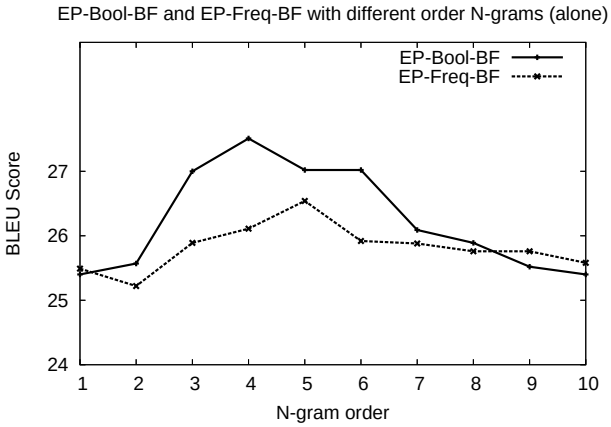


Figure 6: Impact of n-grams of different sizes.

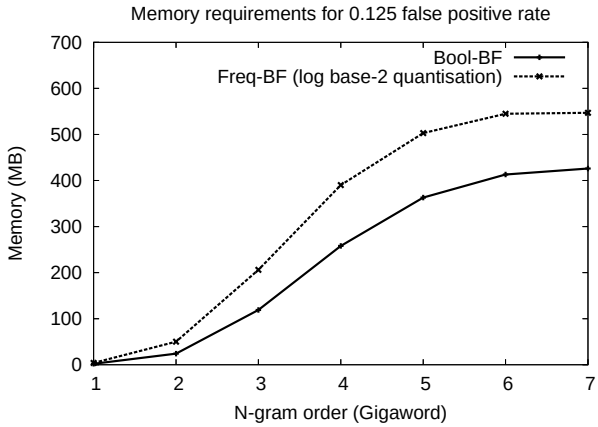


Figure 8: Comparison of memory requirements.

BF for various order n-gram sets from the Gigaword Corpus with the same underlying false positive rate (0.125). The additional space required by our scheme for storing frequency information is less than a factor of 2 compared to the standard BF.

Figure 7 shows the number and size of frequency estimation errors made by our log-frequency BF scheme and a non-redundant scheme that stores only the exact quantised count. We presented 500K negatives to the filter and recorded the frequency of overestimation errors of each size. As shown in Section 3.1, the probability of overestimating an item's frequency under the log-frequency BF scheme decays exponentially in the size of this overestimation error. Although the non-redundant scheme requires fewer items be stored in the filter and, therefore, has a lower underlying false positive rate (0.076 versus 0.159), in practice it incurs a *much* higher error rate (0.717) with many large errors.

Figure 9 shows the impact of sub-sequence filtering on the actual error rate. Although, the false positive rate for the BF containing 2-grams, in addition, to 3-grams (filtered) is higher than the false positive rate of the unfiltered BF containing only 3-grams, the actual error rate of the former is lower for models with less memory. By testing for 2-grams prior to querying for the 3-grams, we can avoid performing some queries that may otherwise have incurred errors using the fact that a 3-gram cannot be present if one of its constituent 2-grams is absent.

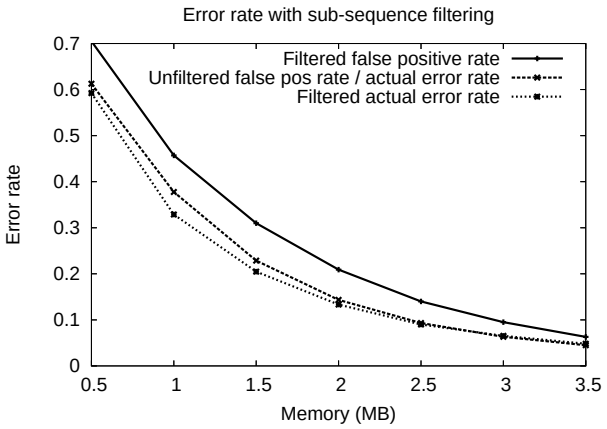


Figure 9: Error rate with sub-sequence filtering.

6 Related Work

We are not the first people to consider building very large scale LMs: Kumar et al. used a four-gram LM for re-ranking (Kumar et al., 2005) and in unpublished work, Google used substantially larger n-grams in their SMT system. Deploying such LMs requires either a cluster of machines (and the overheads of remote procedure calls), per-sentence filtering (which again, is slow) and/or the use of some other lossy compression (Goodman and Gao, 2000). Our approach can complement all these techniques.

Bloom filters have been widely used in database applications for reducing communications overheads and were recently applied to encode word frequencies in information retrieval (Linari and Weikum, 2006) using a method that resembles the non-redundant scheme described above. Extensions of the BF to associate frequencies with items in the set have been proposed e.g., (Cormode and Muthukrishn, 2005); while these schemes are more general than ours, they incur greater space overheads for the distributions that we consider here.

7 Conclusions

We have shown that Bloom Filters can form the basis for space-efficient language modelling in SMT. Extending the standard BF structure to encode corpus frequency information and developing a strategy for reducing the error rates of these models by sub-sequence filtering, our models enable higher-order n-grams and larger monolingual corpora to be used more easily for language modelling in SMT. In a companion paper (Talbot and Osborne, 2007) we have proposed a framework for deriving conventional smoothed n-gram models from the log-frequency BF scheme allowing us to do away entirely with the standard n-gram model in an SMT system. We hope the present work will help establish the Bloom filter as a practical alternative to conventional associative data structures used in computational linguistics. The framework presented here shows that with some consideration for its workings, the randomised nature of the Bloom filter need not be a significant impediment to is use in applications.

References

B. Bloom. 1970. Space/time tradeoffs in hash coding with allowable errors. *CACM*, 13:422–426.

A. Broder and M. Mitzenmacher. 2005. Network applications of bloom filters: A survey. *Internet Mathematics*, 1(4):485–509.

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.

G. Cormode and S. Muthukrishn. 2005. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75.

J. Goodman and J. Gao. 2000. Language model size reduction by pruning and clustering. In *ICSLP'00*, Beijing, China.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *Proc. of the 2007 Conference on Empirical Methods in Natural Language Processing (EMNLP/Co-NLL)*.

P. Koehn. 2003. Europarl: A multilingual corpus for evaluation of machine translation philipp koehn, draft. Available at:http://people.csail.mit.edu/ koehn/publications/europarl.ps.

S. Kumar, Y. Deng, and W. Byrne. 2005. Johns Hopkins University - Cambridge University Chinese-English and Arabic-English 2005 NIST MT Evaluation Systems. In *Proceedings of 2005 NIST MT Workshop*, June.

Alessandro Linari and Gerhard Weikum. 2006. Efficient peer-to-peer semantic overlay networks based on statistical language models. In *Proceedings of the International Workshop on IR in Peer-to-Peer Networks*, pages 9–16, Arlington.

Andreas Stolcke. 2002. SRILM – An extensible language modeling toolkit. In *Proc. of the Intl. Conf. on Spoken Lang. Processing, 2002*.

David Talbot and Miles Osborne. 2007. Smoothed Bloom filter language models: Tera-scale LMs on the cheap. In *Proceedings of the 2007 Conference on Empirical Methods in Natural Language Processing (EMNLP/Co-NLL)*, June.

Bilingual-LSA Based LM Adaptation for Spoken Language Translation

Yik-Cheung Tam and Ian Lane and Tanja Schultz
InterACT, Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
{yct,ian.lane,tanja}@cs.cmu.edu

Abstract

We propose a novel approach to crosslingual language model (LM) adaptation based on bilingual Latent Semantic Analysis (bLSA). A bLSA model is introduced which enables latent topic distributions to be efficiently transferred across languages by enforcing a one-to-one topic correspondence during training. Using the proposed bLSA framework crosslingual LM adaptation can be performed by, first, inferring the topic posterior distribution of the source text and then applying the inferred distribution to the target language N-gram LM via marginal adaptation. The proposed framework also enables rapid bootstrapping of LSA models for new languages based on a source LSA model from another language. On Chinese to English speech and text translation the proposed bLSA framework successfully reduced word perplexity of the English LM by over 27% for a unigram LM and up to 13.6% for a 4-gram LM. Furthermore, the proposed approach consistently improved machine translation quality on both speech and text based adaptation.

1 Introduction

Language model adaptation is crucial to numerous speech and translation tasks as it enables higher-level contextual information to be effectively incorporated into a background LM improving recognition or translation performance. One approach is to employ Latent Semantic Analysis (LSA) to capture in-domain word unigram distributions which are then integrated into the background N-gram LM. This approach has been successfully applied in automatic speech recognition (ASR) (Tam and Schultz, 2006) using the Latent Dirichlet Allocation (LDA) (Blei et al., 2003). The LDA model can be viewed as a Bayesian topic mixture model with the topic mixture weights drawn from a Dirichlet distribution. For LM adaptation, the topic mixture weights are estimated based on in-domain adaptation text (e.g. ASR hypotheses). The adapted mixture weights are then used to interpolate a topic-dependent unigram LM, which is finally integrated into the background N-gram LM using marginal adaptation (Kneser et al., 1997)

In this paper, we propose a framework to perform LM adaptation across languages, enabling the adaptation of a LM from one language based on the adaptation text of another language. In statistical machine translation (SMT), one approach is to apply LM adaptation on the target language based on an initial translation of input references (Kim and Khudanpur, 2003; Paulik et al., 2005). This scheme is limited by the coverage of the translation model, and overall by the quality of translation. Since this approach only allows to apply LM adaptation *after* translation, available knowledge cannot be applied to extend the coverage. We propose a bilingual LSA model (bLSA) for crosslingual LM adaptation that can be applied *before* translation. The bLSA model consists of two LSA models: one for each side of the language trained on parallel document corpora. The key property of the bLSA model is that

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 520–527,
Prague, Czech Republic, June 2007.

the latent topic of the source and target LSA models can be assumed to be a one-to-one correspondence and thus share a common latent topic space since the training corpora consist of bilingual parallel data. For instance, say topic 10 of the Chinese LSA model is about politics. Then topic 10 of the English LSA model is set to also correspond to politics and so forth. During LM adaptation, we first infer the topic mixture weights from the source text using the source LSA model. Then we transfer the inferred mixture weights to the target LSA model and thus obtain the target LSA marginals. The challenge is to enforce the one-to-one topic correspondence. Our proposal is to share common variational Dirichlet posteriors over the topic mixture weights of a document pair in the LDA-style model. The beauty of the bLSA framework is that the model searches for a common latent topic space in an unsupervised fashion, rather than to require manual interaction. Since the topic space is language independent, our approach supports topic transfer in multiple language pairs in O(N) where N is the number of languages.

Related work includes the Bilingual Topic Admixture Model (BiTAM) for word alignment proposed by (Zhao and Xing, 2006). Basically, the BiTAM model consists of topic-dependent translation lexicons modeling $Pr(c|e,k)$ where c, e and k denotes the source Chinese word, target English word and the topic index respectively. On the other hand, the bLSA framework models $Pr(c|k)$ and $Pr(e|k)$ which is different from the BiTAM model. By their different modeling nature, the bLSA model usually supports more topics than the BiTAM model. Another work by (Kim and Khudanpur, 2004) employed crosslingual LSA using singular value decomposition which concatenates bilingual documents into a single input supervector before projection.

We organize the paper as follows: In Section 2, we introduce the bLSA framework including Latent Dirichlet-Tree Allocation (LDTA) (Tam and Schultz, 2007) as a correlated LSA model, bLSA training and crosslingual LM adaptation. In Section 3, we present the effect of LM adaptation on word perplexity, followed by SMT experiments reported in BLEU on both speech and text input in Section 3.3. Section 4 describes conclusions and future works.

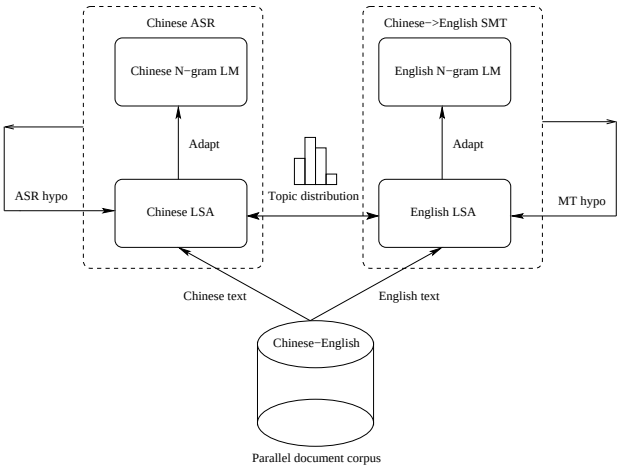


Figure 1: Topic transfer in bilingual LSA model.

2 Bilingual Latent Semantic Analysis

The goal of a bLSA model is to enforce a one-to-one topic correspondence between monolingual LSA models, each of which can be modeled using an LDA-style model. The role of the bLSA model is to transfer the inferred latent topic distribution from the source language to the target language assuming that the topic distributions on both sides are identical. The assumption is reasonable for parallel document pairs which are faithful translations. Figure 1 illustrates the idea of topic transfer between monolingual LSA models followed by LM adaptation. One observation is that the topic transfer can be bi-directional meaning that the "flow" of topic can be from ASR to SMT or vice versa. In this paper, we only focus on ASR-to-SMT direction. Our target is to minimize the word perplexity on the target language through LM adaptation. Before we introduce the heuristic of enforcing a one-to-one topic correspondence, we describe the Latent Dirichlet-Tree Allocation (LDTA) for LSA.

2.1 Latent Dirichlet-Tree Allocation

The LDTA model extends the LDA model in which correlation among latent topics are captured using a Dirichlet-Tree prior. Figure 2 illustrates a depth-two Dirichlet-Tree. A tree of depth one simply falls back to the LDA model. The LDTA model is a generative model with the following generative process:

1. Sample a vector of branch probabilities $b_j \sim$

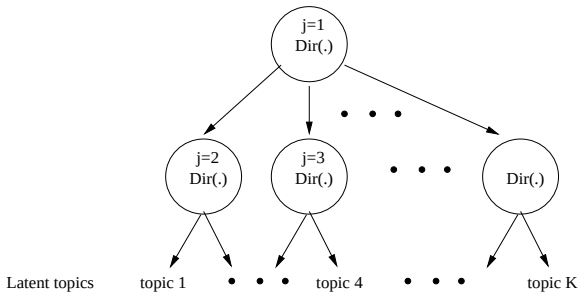


Figure 2: Dirichlet-Tree prior of depth two.

$Dir(\alpha_j)$ for each node $j = 1...J$ where α_j denotes the parameter (aka the pseudo-counts of its outgoing branches) of the Dirichlet distribution at node j.

2. Compute the topic proportions as:

$$\theta_k = \prod_{jc} b_{jc}^{\delta_{jc}(k)} \quad (1)$$

where $\delta_{jc}(k)$ is an indicator function which sets to unity when the c-th branch of the j-th node leads to the leaf node of topic k and zero otherwise. The k-th topic proportion θ_k is computed as the product of branch probabilities from the root node to the leaf node of topic k.

3. Generate a document using the topic multinomial for each word w_i:

$$z_i \sim Mult(\theta)$$
$$w_i \sim Mult(\beta_{.z_i})$$

where $\beta_{.z_i}$ denotes the topic-dependent unigram LM indexed by z_i.

The joint distribution of the latent variables (topic sequence z_1^n and the Dirichlet nodes over child branches b_j) and an observed document w_1^n can be written as follows:

$$p(w_1^n, z_1^n, b_1^J) = p(b_1^J|\{\alpha_j\}) \prod_i^n \beta_{w_i z_i} \cdot \theta_{z_i}$$

$$\text{where} \quad p(b_1^J|\{\alpha_j\}) = \prod_j^J Dir(b_j; \alpha_j) \propto \prod_{jc} b_{jc}^{\alpha_{jc}-1}$$

Similar to LDA training, we apply the variational Bayes approach by optimizing the lower bound of the marginalized document likelihood:

$$L(w_1^n; \Lambda, \Gamma) = E_q[\log \frac{p(w_1^n, z_1^n, b_1^J; \Lambda)}{q(z_1^n, b_1^J; \Gamma)}]$$
$$= E_q[\log p(w_1^n|z_1^n)] + E_q[\log \frac{p(z_1^n|b_1^J)}{q(z_1^n)}]$$
$$+ E_q[\log \frac{p(b_1^J; \{\alpha_j\})}{q(b_1^J; \{\gamma_j\})}]$$

where $q(z_1^n, b_1^J; \Gamma) = \prod_i^n q(z_i) \cdot \prod_j^J q(b_j)$ is a factorizable variational posterior distribution over the latent variables parameterized by Γ which are determined in the E-step. Λ is the model parameters for a Dirichlet-Tree $\{\alpha_j\}$ and the topic-dependent unigram LM $\{\beta_{wk}\}$. The LDTA model has an E-step similar to the LDA model:

E-Step:

$$\gamma_{jc} = \alpha_{jc} + \sum_i^n \sum_k^K q_{ik} \cdot \delta_{jc}(k) \quad (2)$$

$$q_{ik} \propto \beta_{w_i k} \cdot e^{E_q[\log \theta_k]} \quad (3)$$

where

$$E_q[\log \theta_k] = \sum_{jc} \delta_{jc}(k) E_q[\log b_{jc}]$$
$$= \sum_{jc} \delta_{jc}(k) \left(\Psi(\gamma_{jc}) - \Psi(\sum_c \gamma_{jc}) \right)$$

where q_{ik} denotes $q(z_i = k)$ meaning the variational topic posterior of word w_i. Eqn 2 and Eqn 3 are executed iteratively until convergence is reached.

M-Step:

$$\beta_{wk} \propto \sum_i^n q_{ik} \cdot \delta(w_i, w) \quad (4)$$

where $\delta(w_i, w)$ is a Kronecker Delta function. The alpha parameters can be estimated with iterative methods such as Newton-Raphson or simple gradient ascent procedure.

2.2 Bilingual LSA training

For the following explanations, we assume that our source and target languages are Chinese and English respectively. The bLSA model training is a

two-stage procedure. At the first stage, we train a Chinese LSA model using the Chinese documents in parallel corpora. We applied the variational EM algorithm (Eqn 2–4) to train a Chinese LSA model. Then we used the model to compute the term $e^{E_q[\log \theta_k]}$ needed in Eqn 3 for each Chinese document in parallel corpora. At the second stage, we apply the same $e^{E_q[\log \theta_k]}$ to *bootstrap* an English LSA model, which is the key to enforce a one-to-one topic correspondence. Now the hyper-parameters of the variational Dirichlet posteriors of each node in the Dirichlet-Tree are shared among the Chinese and English model. Precisely, we apply only Eqn 3 with fixed $e^{E_q[\log \theta_k]}$ in the E-step and Eqn 4 in the M-step on $\{\beta_{wk}\}$ to bootstrap an English LSA model. Notice that the E-step is non-iterative resulting in rapid LSA training. In short, given a monolingual LSA model, we can rapidly bootstrap LSA models of new languages using parallel document corpora. Notice that the English and Chinese vocabulary sizes do not need to be similar. In our setup, the Chinese vocabulary comes from the ASR system while the English vocabulary comes from the English part of the parallel corpora. Since the topic transfer can be bi-directional, we can perform the bLSA training in a reverse manner, i.e. training an English LSA model followed by bootstrapping a Chinese LSA model.

2.3 Crosslingual LM adaptation

Given a source text, we apply the E-step to estimate variational Dirichlet posterior of each node in the Dirichlet-Tree. We estimate the topic weights on the source language using the following equation:

$$\hat{\theta}_k^{(CH)} \propto \prod_{jc} \left(\frac{\gamma_{jc}}{\sum_{c'} \gamma_{jc'}} \right)^{\delta_{jc}(k)} \quad (5)$$

Then we apply the topic weights into the target LSA model to obtain an in-domain LSA marginals:

$$Pr_{EN}(w) = \sum_{k=1}^{K} \beta_{wk}^{(EN)} \cdot \hat{\theta}_k^{(CH)} \quad (6)$$

We integrate the LSA marginal into the target background LM using marginal adaptation (Kneser et al., 1997) which minimizes the Kullback-Leibler divergence between the adapted LM and the background LM:

$$Pr_a(w|h) \propto \left(\frac{Pr_{ldta}(w)}{Pr_{bg}(w)} \right)^{\beta} \cdot Pr_{bg}(w|h) \quad (7)$$

Likewise, LM adaptation can take place on the source language as well due to the bi-directional nature of the bLSA framework when target-side adaptation text is available. In this paper, we focus on LM adaptation on the target language for SMT.

3 Experimental Setup

We evaluated our bLSA model using the Chinese–English parallel document corpora consisting of the Xinhua news, Hong Kong news and Sina news. The combined corpora contains 67k parallel documents with 35M Chinese (CH) words and 43M English (EN) words. Our spoken language translation system translates from Chinese to English. The Chinese vocabulary comes from the ASR decoder while the English vocabulary is derived from the English portion of the parallel training corpora. The vocabulary sizes for Chinese and English are 108k and 69k respectively. Our background English LM is a 4-gram LM trained with the modified Kneser-Ney smoothing scheme using the SRILM toolkit on the same training text. We explore the bLSA training in both directions: EN→CH and CH→EN meaning that an English LSA model is trained first and a Chinese LSA model is bootstrapped or vice versa. Experiments explore which bootstrapping direction yield best results measured in terms of English word perplexity. The number of latent topics is set to 200 and a balanced binary Dirichlet-Tree prior is used.

With an increasing interest in the ASR-SMT coupling for spoken language translation, we also evaluated our approach with Chinese ASR hypotheses and compared with Chinese manual transcriptions. We are interested to see the impact due to recognition errors on the ASR hypotheses compared to the manual transcriptions. We employed the CMU-InterACT ASR system developed for the GALE 2006 evaluation. We trained acoustic models with over 500 hours of quickly transcribed speech data released by the GALE program and the LM with over 800M-word Chinese corpora. The character error rates on the CCTV, RFA and NTDTV shows in the RT04 test set are 7.4%, 25.5% and 13.1% respectively.

Topic index	Top words
"CH-40"	flying, submarine, aircraft, air, pilot, land, mission, brand-new
"EN-40"	air, sea, submarine, aircraft, flight, flying, ship, test
"CH-41"	satellite, han-tian, launch, space, china, technology, astronomy
"EN-41"	space, satellite, china, technology, satellites, science
"CH-42"	fire, airport, services, marine, accident, air
"EN-42"	fire, airport, services, department, marine, air, service

Table 1: Parallel topics extracted by the bLSA model. Top words on the Chinese side are translated into English for illustration purpose.

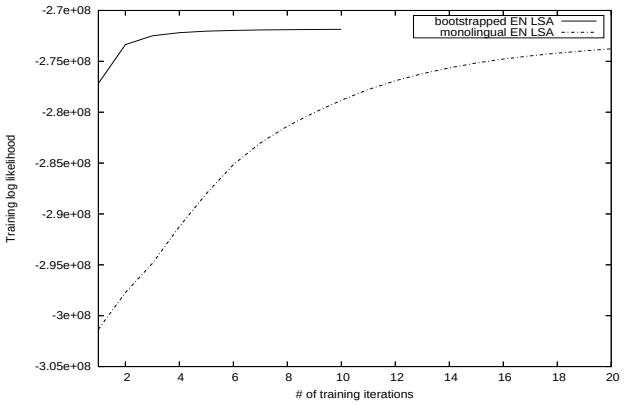


Figure 3: Comparison of training log likelihood of English LSA models bootstrapped from a Chinese LSA and from a flat monolingual English LSA.

3.1 Analysis of the bLSA model

By examining the top-words of the extracted parallel topics, we verify the validity of the heuristic described in Section 2.2 which enforces a one-to-one topic correspondence in the bLSA model. Table 1 shows the latent topics extracted by the CH→EN bLSA model. We can see that the Chinese-English topic words have strong correlations. Many of them are actually translation pairs with similar word rankings. From this viewpoint, we can interpret bLSA as a crosslingual word trigger model. The result indicates that our heuristic is effective to extract parallel latent topics. As a sanity check, we also examine the likelihood of the training data when an English LSA model is bootstrapped. We can see from Figure 3 that the likelihood increases monotonically with the number of training iterations. The figure also shows that by sharing the variational Dirichlet posteriors from the Chinese LSA model, we can bootstrap an English LSA model *rapidly* compared to monolingual English LSA training with both training procedures started from the same flat model.

LM (43M)	CCTV	RFA	NTDTV
BG EN unigram	1065	1220	1549
+CH→EN (CH ref)	755	880	1113
+EN→CH (CH ref)	762	896	1111
+CH→EN (CH hypo)	757	885	1126
+EN→CH (CH hypo)	766	896	1129
+CH→EN (EN ref)	731	838	1075
+EN→CH (EN ref)	747	848	1087

Table 2: English word perplexity (PPL) on the RT04 test set using a unigram LM.

3.2 LM adaptation results

We trained the bLSA models on both CH→EN and EN→CH directions and compared their LM adaptation performance using the Chinese ASR hypotheses (hypo) and the manual transcriptions (ref) as input. We adapted the English background LM using the LSA marginals described in Section 2.3 for each show on the test set.

We first evaluated the English word perplexity using the EN unigram LM generated by the bLSA model. Table 2 shows that the bLSA-based LM adaptation reduces the word perplexity by over 27% relative compared to an unadapted EN unigram LM. The results indicate that the bLSA model successfully leverages the text from the source language and improves the word perplexity on the target language. We observe that there is almost no performance difference when either the ASR hypotheses or the manual transcriptions are used for adaptation. The result is encouraging since the bLSA model may be insensitive to moderate recognition errors through the projection of the input adaptation text into the latent topic space. We also apply an English translation reference for adaptation to show an oracle performance. The results using the Chinese hypotheses are not too far off from the oracle performance. Another observation is that the CH→EN bLSA model seems to give better performance than the EN→CH bLSA model. However, their differences are not significant. The result may imply that the direction of the bLSA training is not important since the latent topic space captured by either language is similar when parallel training corpora are used. Table 3 shows the word perplexity when the background 4-gram English LM is adapted with the tuning parameter β set

LM (43M, $\beta = 0.7$)	CCTV	RFA	NTDTV
BG EN 4-gram	118	212	203
+CH→EN (CH ref)	102	191	179
+EN→CH (CH ref)	102	198	179
+CH→EN (CH hypo)	102	193	180
+EN→CH (CH hypo)	103	198	180
+CH→EN (EN ref)	100	186	176
+EN→CH (EN ref)	101	190	176

Table 3: English word perplexity (PPL) on the RT04 test set using a 4-gram LM.

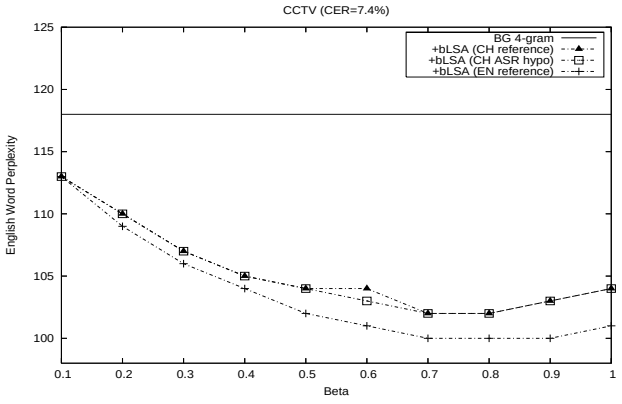


Figure 4: Word perplexity with different β using manual reference or ASR hypotheses on CCTV.

to 0.7. Figure 4 shows the change of perplexity with different β. We see that the adaptation performance using the ASR hypotheses or the manual transcriptions are almost identical on different β with an optimal value at around 0.7. The results show that the proposed approach successfully reduces the perplexity in the range of 9–13.6% relative compared to an unadapted baseline on different shows when ASR hypotheses are used. Moreover, we observe similar performance using ASR hypotheses or manual Chinese transcriptions which is consistent with the results on Table 2. On the other hand, it is interesting to see that the performance gap from the oracle adaptation is somewhat related to the degree of mismatch between the test show and the training condition. The gap looks wider on the RFA and NTDTV shows compared to the CCTV show.

3.3 Incorporating bLSA into Spoken Language Translation

To investigate the effectiveness of bLSA LM adaptation for spoken language translation, we incorporated the proposed approach into our state-of-the-art phrase-based SMT system. Translation performance was evaluated on the RT04 broadcast news evaluation set when applied to both the manual transcriptions and 1-best ASR hypotheses. During evaluation two performance metrics, BLEU (Papineni et al., 2002) and NIST, were computed. In both cases, a single English reference was used during scoring. In the transcription case the original English references were used. For the ASR case, as utterance segmentation was performed automatically, the number of sentences generated by ASR and SMT differed from the number of English references. In this case, Levenshtein alignment was used to align the translation output to the English references before scoring.

3.4 Baseline SMT Setup

The baseline SMT system consisted of a non adaptive system trained using the same Chinese-English parallel document corpora used in the previous experiments (Sections 3.1 and 3.2). For phrase extraction a cleaned subset of these corpora, consisting of 1M Chinese-English sentence pairs, was used. SMT decoding parameters were optimized using manual transcriptions and translations of 272 utterances from the RT04 development set (LDC2006E10).

SMT translation was performed in two stages using an approach similar to that in (Vogel, 2003). First, a translation lattice was constructed by matching all possible bilingual phrase-pairs, extracted from the training corpora, to the input sentence. Phrase extraction was performed using the "*PESA*" (Phrase Pair Extraction as Sentence Splitting) approach described in (Vogel, 2005). Next, a search was performed to find the best path through the lattice, i.e. that with maximum *translation-score*. During search reordering was allowed on the target language side. The final translation result was that hypothesis with maximum *translation-score*, which is a log-linear combination of 10 scores consisting of Target LM probability, Distortion Penalty, Word-Count Penalty, Phrase-Count and six Phrase-Alignment scores. Weights for each component score were optimized to maximize BLEU-score on the development set using MER optimization as described in (Venugopal et al., 2005).

SMT Target LM	Translation Quality - BLEU (NIST) CCTV	RFA	NTDTV	ALL
	Manual Transcription			
Baseline LM:	0.162 (5.212)	0.087 (3.854)	0.140 (4.859)	0.132 (5.146)
bLSA (bLSA-Adapted LM):	0.164 (5.212)	0.087 (3.897)	0.143 (4.864)	0.134 (5.162)
	1-best ASR Output			
CER (%)	7.4	25.5	13.1	14.9
Baseline LM:	0.129 (4.15)	0.051 (2.77)	0.086 (3.50)	0.095 (3.90)
bLSA (bLSA-Adapted LM):	0.132 (4.16)	0.050 (2.79)	0.089 (3.53)	0.096 (3.91)

Table 4: Translation performance of baseline and bLSA-Adapted Chinese-English SMT systems on manual transcriptions and 1-best ASR hypotheses

3.5 Performance of Baseline SMT System

First, the baseline system performance was evaluated by applying the system described above to the reference transcriptions and 1-best ASR hypotheses generated by our Mandarin speech recognition system. The translation accuracy in terms of BLEU and NIST for each individual show ("*CCTV*", "*RFA*", and "*NTDTV*"), and for the complete test-set, are shown in Table 4 (**Baseline LM**). When applied to the reference transcriptions an overall BLEU score of 0.132 was obtained. BLEU-scores ranged between 0.087 and 0.162 for the "*RFA*", "*NTDTV*" and "*CCTV*" shows, respectively. As the "*RFA*" show contained a large segment of conversational speech, translation quality was considerably lower for this show due to genre mismatch with the training corpora of newspaper text.

For the 1-best ASR hypotheses, an overall BLEU score of 0.095 was achieved. For the ASR case, the relative reduction in BLEU scores for the RFA and NTDTV shows is large, due to the significantly lower recognition accuracies for these shows. BLEU score is also degraded due to poor alignment of references during scoring.

3.6 Incorporation of bLSA Adaptation

Next, the effectiveness of bLSA based LM adaptation was evaluated. For each show the target English LM was adapted using bLSA-adaptation, as described in Section 2.3. SMT was then applied using an identical setup to that used in the baseline experiments.

The translation accuracy when bLSA adaptation was incorporated is shown in Table 4. When applied to the manual transcriptions, bLSA adaptation improved the overall BLEU-score by 1.7% relative (from 0.132 to 0.134). For all three shows bLSA adaptation gained higher BLEU and NIST metrics. A similar trend was also observed when the proposed approach was applied to the 1-best ASR output. On the evaluation set a relative improvement in BLEU score of 1.0% was gained.

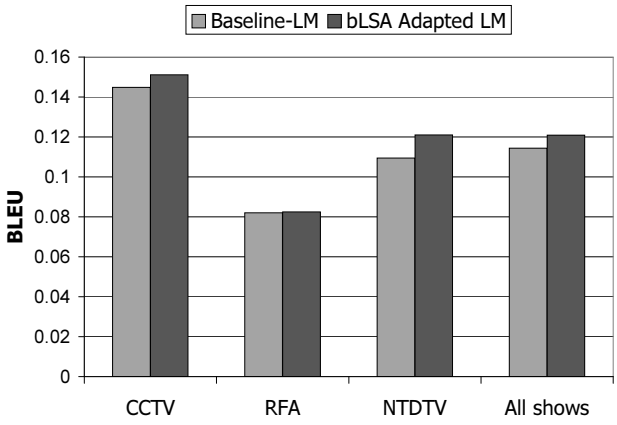


Figure 5: BLEU score for those 25% utterances which resulted in different translations after bLSA adaptation (manual transcriptions)

The semantic interpretation of the majority of utterances in broadcast news are not affected by topic context. In the experimental evaluation it was observed that only 25% of utterances produced different translation output when bLSA adaptation was performed compared to the topic-independent baseline. Although the improvement in translation quality (BLEU) was small when evaluated over the entire test set, the improvement in BLEU score for

these 25% utterances was significant. The translation quality for the baseline and bLSA-adaptive system when evaluated only on these utterances is shown in Figure 5 for the manual transcription case. On this subset of utterances an overall improvement in BLEU of 0.007 (5.7% relative) was gained, with a gain of 0.012 (10.6% relative) points for the "*NTDTV*" show. A similar trend was observed when applied to the 1-best ASR output. In this case a relative improvement in BLEU of 12.6% was gained for "*NTDTV*", and for "All shows" 0.007 (3.7%) was gained. Current evaluation metrics for translation, such as "BLEU", do not consider the relative importance of specific words or phrases during translation and thus are unable to highlight the true effectiveness of the proposed approach. In future work, we intend to investigate other evaluation metrics which consider the relative informational content of words.

4 Conclusions

We proposed a bilingual latent semantic model for crosslingual LM adaptation in spoken language translation. The bLSA model consists of a set of monolingual LSA models in which a one-to-one topic correspondence is enforced between the LSA models through the sharing of variational Dirichlet posteriors. Bootstrapping a LSA model for a new language can be performed rapidly with topic transfer from a well-trained LSA model of another language. We transfer the inferred topic distribution from the input source text to the target language effectively to obtain an in-domain target LSA marginals for LM adaptation. Results showed that our approach significantly reduces the word perplexity on the target language in both cases using ASR hypotheses and manual transcripts. Interestingly, the adaptation performance is not much affected when ASR hypotheses were used. We evaluated the adapted LM on SMT and found that the evaluation metrics are crucial to reflect the actual improvement in performance. Future directions include the exploration of story-dependent LM adaptation with automatic story segmentation instead of show-dependent adaptation due to the possibility of multiple stories within a show. We will investigate the incorporation of monolingual documents for potentially better bilingual LSA modeling.

Acknowledgment

This work is partly supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-2-0001. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

D. Blei, A. Ng, and M. Jordan. 2003. Latent Dirichlet Allocation. In *Journal of Machine Learning Research*, pages 1107–1135.

W. Kim and S. Khudanpur. 2003. LM adaptation using cross-lingual information. In *Proc. of Eurospeech*.

W. Kim and S. Khudanpur. 2004. Cross-lingual latent semantic analysis for LM. In *Proc. of ICASSP*.

R. Kneser, J. Peters, and D. Klakow. 1997. Language model adaptation using dynamic marginals. In *Proc. of Eurospeech*, pages 1971–1974.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proc. of ACL*.

M. Paulik, C. Fügen, T. Schaaf, T. Schultz, S. Stüker, and A. Waibel. 2005. Document driven machine translation enhanced automatic speech recognition. In *Proc. of Interspeech*.

Y. C. Tam and T. Schultz. 2006. Unsupervised language model adaptation using latent semantic marginals. In *Proc. of Interspeech*.

Y. C. Tam and T. Schultz. 2007. Correlated latent semantic model for unsupervised language model adaptation. In *Proc. of ICASSP*.

A. Venugopal, A. Zollmann, and A. Waibel. 2005. Training and evaluation error minimization rules for statistical machine translation. In *Proc. of ACL*.

S. Vogel. 2003. SMT decoder dissected: Word reordering. In *Proc. of ICNLPKE*.

S. Vogel. 2005. PESA: Phrase pair extraction as sentence splitting. In *Proc. of the Machine Translation Summit*.

B. Zhao and E. P. Xing. 2006. BiTAM: Bilingual topic admixture models for word alignment. In *Proc. of ACL*.

Coreference Resolution Using Semantic Relatedness Information from Automatically Discovered Patterns

Xiaofeng Yang **Jian Su**
Institute for Infocomm Research
21 Heng Mui Keng Terrace, Singapore, 119613
{xiaofengy,sujian}@i2r.a-star.edu.sg

Abstract

Semantic relatedness is a very important factor for the coreference resolution task. To obtain this semantic information, corpus-based approaches commonly leverage patterns that can express a specific semantic relation. The patterns, however, are designed manually and thus are not necessarily the most effective ones in terms of accuracy and breadth. To deal with this problem, in this paper we propose an approach that can automatically find the effective patterns for coreference resolution. We explore how to automatically discover and evaluate patterns, and how to exploit the patterns to obtain the semantic relatedness information. The evaluation on ACE data set shows that the pattern based semantic information is helpful for coreference resolution.

1 Introduction

Semantic relatedness is a very important factor for coreference resolution, as noun phrases used to refer to the same entity should have a certain semantic relation. To obtain this semantic information, previous work on reference resolution usually leverages a semantic lexicon like WordNet (Vieira and Poesio, 2000; Harabagiu et al., 2001; Soon et al., 2001; Ng and Cardie, 2002). However, the drawback of WordNet is that many expressions (especially for proper names), word senses and semantic relations are not available from the database (Vieira and Poesio, 2000). In recent years, increasing interest has been seen in mining semantic relations from large text corpora. One common solution is to utilize a pattern that can represent a specific semantic relation (e.g., "*X such as Y*" for *is-a* relation, and "*X and other Y*" for *other*-relation). Instantiated with two given noun phrases, the pattern is searched in a large corpus and the occurrence number is used as a measure of their semantic relatedness (Markert et al., 2003; Modjeska et al., 2003; Poesio et al., 2004).

However, in the previous pattern based approaches, the selection of the patterns to represent a specific semantic relation is done in an ad hoc way, usually by linguistic intuition. The manually selected patterns, nevertheless, are not necessarily the most effective ones for coreference resolution from the following two concerns:

- *Accuracy*. Can the patterns (e.g., "*X such as Y*") find as many NP pairs of the specific semantic relation (e.g. *is-a*) as possible, with a high precision?
- *Breadth*. Can the patterns cover a wide variety of semantic relations, not just *is-a*, by which coreference relationship is realized? For example, in some annotation schemes like ACE, "Beijing:China" are coreferential as the capital and the country could be used to represent the government. The pattern for the common "is-a" relation will fail to identify the NP pairs of such a "capital-country" relation.

To deal with this problem, in this paper we propose an approach which can automatically discover effective patterns to represent the semantic relations

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 528–535,
Prague, Czech Republic, June 2007.

for coreference resolution. We explore two issues in our study:

(1) *How to automatically acquire and evaluate the patterns?* We utilize a set of coreferential NP pairs as seeds. For each seed pair, we search a large corpus for the texts where the two noun phrases co-occur, and collect the surrounding words as the surface patterns. We evaluate a pattern based on its commonality or association with the positive seed pairs.

(2) *How to mine the patterns to obtain the semantic relatedness information for coreference resolution?* We present two strategies to exploit the patterns: choosing the top best patterns as a set of pattern features, or computing the reliability of semantic relatedness as a single feature. In either strategy, the obtained features are applied to do coreference resolution in a supervised-learning way.

To our knowledge, our work is the first effort that systematically explores these issues in the coreference resolution task. We evaluate our approach on ACE data set. The experimental results show that the pattern based semantic relatedness information is helpful for the coreference resolution.

The remainder of the paper is organized as follows. Section 2 gives some related work. Section 3 introduces the framework for coreference resolution. Section 4 presents the model to obtain the pattern-based semantic relatedness information. Section 5 discusses the experimental results. Finally, Section 6 summarizes the conclusions.

2 Related Work

Earlier work on coreference resolution commonly relies on semantic lexicons for semantic relatedness knowledge. In the system by Vieira and Poesio (2000), for example, WordNet is consulted to obtain the synonymy, hypernymy and meronymy relations for resolving the definite anaphora. In (Harabagiu et al., 2001), the path patterns in WordNet are utilized to compute the semantic consistency between NPs. Recently, Ponzetto and Strube (2006) suggest to mine semantic relatedness from Wikipedia, which can deal with the data sparseness problem suffered by using WordNet.

Instead of leveraging existing lexicons, many researchers have investigated corpus-based approaches to mine semantic relations. Garera and Yarowsky (2006) propose an unsupervised model which extracts hypernym relation for resloving definite NPs. Their model assumes that a definite NP and its hypernym words usually co-occur in texts. Thus, for a definite-NP anaphor, a preceding NP that has a high co-occurrence statistics in a large corpus is preferred for the antecedent.

Bean and Riloff (2004) present a system called BABAR that uses contextual role knowledge to do coreference resolution. They apply an IE component to unannotated texts to generate a set of extraction caseframes. Each caseframe represents a linguistic expression and a syntactic position, e.g. "murder of <NP>", "killed <patient>". From the caseframes, they derive different types of contextual role knowledge for resolution, for example, whether an anaphor and an antecedent candidate can be filled into co-occurring caseframes, or whether they are substitutable for each other in their caseframes. Different from their system, our approach aims to find surface patterns that can directly indicate the coreference relation between two NPs.

Hearst (1998) presents a method to automate the discovery of WordNet relations, by searching for the corresponding patterns in large text corpora. She explores several patterns for the hyponymy relation, including "*X such as Y*" "*X and/or other Y*", "*X including / especially Y*" and so on. The use of Hearst's style patterns can be seen for the reference resolution task. Modjeska et al. (2003) explore the use of the Web to do the *other*-anaphora resolution. In their approach, a pattern "*X and other Y*" is used. Given an anaphor and a candidate antecedent, the pattern is instantiated with the two NPs and forms a query. The query is submitted to the Google searching engine, and the returned hit number is utilized to compute the semantic relatedness between the two NPs. In their work, the semantic information is used as a feature for the learner. Markert et al. (2003) and Poesio et al. (2004) adopt a similar strategy for the *bridging* anaphora resolution.

In (Hearst, 1998), the author also proposes to discover new patterns instead of using the manually designed ones. She employs a bootstrapping algorithm to learn new patterns from the word pairs with a known relation. Based on Hearst's work, Pantel and Pennacchiotti (2006) further give a method

which measures the reliability of the patterns based on the strength of association between patterns and instances, employing the pointwise mutual information (PMI).

3 Framework of Coreference Resolution

Our coreference resolution system adopts the common learning-based framework as employed by Soon et al. (2001) and Ng and Cardie (2002).

In the learning framework, a training or testing instance has the form of $\boldsymbol{i}\{NP_i, NP_j\}$, in which NP_j is a possible anaphor and NP_i is one of its antecedent candidates. An instance is associated with a vector of features, which is used to describe the properties of the two noun phrases as well as their relationships. In our baseline system, we adopt the common features for coreference resolution such as lexical property, distance, string-matching, name-alias, apposition, grammatical role, number/gender agreement and so on. The same feature set is described in (Ng and Cardie, 2002) for reference.

During training, for each encountered anaphor NP_j, one single positive training instance is created for its closest antecedent. And a group of negative training instances is created for every intervening noun phrases between NP_j and the antecedent.

Based on the training instances, a binary classifier can be generated using any discriminative learning algorithm, like C5 in our study. For resolution, an input document is processed from the first NP to the last. For each encountered NP_j, a test instance is formed for each antecedent candidate, NP_i[1]. This instance is presented to the classifier to determine the coreference relationship. NP_j will be resolved to the candidate that is classified as positive (if any) and has the highest confidence value.

In our study, we augment the common framework by incorporating non-anaphors into training. We focus on the non-anaphors that the original classifier fails to identify. Specifically, we apply the learned classifier to all the non-anaphors in the training documents. For each non-anaphor that is classified as positive, a negative instance is created by pairing the non-anaphor and its false antecedent. These negative instances are added into the original training instance set for learning, which will generate a classifier with the capability of not only antecedent identification, but also non-anaphorically identification. The new classier is applied to the testing document to do coreference resolution as usual.

[1]For resolution of pronouns, only the preceding NPs in current and previous two sentences are considered as antecedent candidates. For resolution of non-pronouns, all the preceding non-pronouns are considered.

4 Patterned Based Semantic Relatedness

4.1 Acquiring the Patterns

To derive patterns to indicate a specific semantic relation, a set of seed NP pairs that have the relation of interest is needed. As described in the previous section, we have a set of training instances formed by NP pairs with known coreference relationships. We can just use this set of NP pairs as the seeds. That is, an instance $\boldsymbol{i}\{NP_i, NP_j\}$ will become a seed pair (E_i:E_j) in which NP_i corresponds to E_i and NP_j corresponds to E_j. In creating the seed, for a common noun, only the head word is retained while for a proper name, the whole string is kept. For example, instance $\boldsymbol{i}$\{"Bill Clinton", "the former president"\} will be converted to a NP pair ("Bill Clinton":"president").

We create the seed pair for every training instance $\boldsymbol{i}\{NP_i, NP_j\}$, except when (1) NP_i or NP_j is a pronoun; or (2) NP_i and NP_j have the same head word. We denote S+ and S- the set of seed pairs derived from the positive and the negative training instances, respectively. Note that a seed pair may possibly belong to S+ can S- at the same time.

For each of the seed NP pairs (E_i:E_j), we search in a large corpus for the strings that match the regular expression "E_i * * * E_j" or "E_j * * * E_i", where * is a wildcard for any word or symbol. The regular expression is defined as such that all the co-occurrences of E_i and E_j with at most three words (or symbols) in between are retrieved.

For each retrieved string, we extract a surface pattern by replacing expression E_i with a mark <#t1#> and E_j with <#t2#>. If the string is followed by a symbol, the symbol will be also included in the pattern. This is to create patterns like "X * * * Y [, . ?]" where *Y*, with a high possibility, is the head word, but not a modifier of another noun phrase.

As an example, consider the pair ("Bill Clinton":"president"). Suppose that two sentences in a corpus can be matched by the regular expressions:

(S1) " ***Bill Clinton*** *is elected* ***President*** *of the United States.*"

(S2) "*The US* ***President****, Mr* ***Bill Clinton****, today advised India to move towards nuclear non-proliferation and begin a dialogue with Pakistan to ...* ".

The patterns to be extracted for (S1) and (S2), respectively, are

P1: <#t1#> ***is elected*** <#t2#>

P2: <#t2#> **, *Mr*** <#t1#> **,**

We record the number of strings matched by a pattern p instantiated with (E_i:E_j), noted $|(E_i, p, E_j)|$, for later use.

For each seed pair, we generate a list of surface patterns in the above way. We collect all the patterns derived from the positive seed pairs as a set of reference patterns, which will be scored and used to evaluate the semantic relatedness for any new NP pair.

4.2 Scoring the Patterns

4.2.1 Frequency

One possible scoring scheme is to evaluate a pattern based on its commonality to positive seed pairs. The intuition here is that the more often a pattern is seen for the positive seed pairs, the more indicative the pattern is to find positive coreferential NP pairs. Based on this idea, we score a pattern by calculating the number of positive seed pairs whose pattern list contains the pattern. Formally, supposing the pattern list associated with a seed pair s is PList(s), the frequency score of a pattern p is defined as

$$Freqency(p) = |\{s|s \in S+, p \in PList(s)\}| \quad (1)$$

4.2.2 Reliability

Another possible way to evaluate a pattern is based on its reliability, i.e., the degree that the pattern is associated with the positive coreferential NPs. In our study, we use pointwise mutual information (Cover and Thomas, 1991) to measure association strength, which has been proved effective in the task of semantic relation identification (Pantel and Pennacchiotti, 2006). Under pointwise mutual information (PMI), the strength of association between two events x and y is defined as follows:

$$pmi(x, y) = \log \frac{P(x, y)}{P(x)P(y)} \quad (2)$$

Thus the association between a pattern p and a positive seed pair s:(E_i:E_j) is:

$$pmi(p, (E_i : E_j)) = \log \frac{\frac{|(E_i,p,E_j)|}{|(*,*,*)|}}{\frac{|(E_i,*,E_j)|}{|(*,*,*)|}\frac{|(*,p,*)|}{|(*,*,*)|}} \quad (3)$$

where $|(E_i$,p,$E_j)|$ is the count of strings matched by pattern p instantiated with E_i and E_j. Asterisk * represents a wildcard, that is:

$$|(E_i, *, E_j)| = \sum_{p \in PList(E_i:E_j)} |(E_i, p, E_j)| \quad (4)$$

$$|(*, p, *)| = \sum_{(E_i:E_j) \in S+ \cup S-} |(E_i, p, E_j)| \quad (5)$$

$$|(*, *, *)| = \sum_{(E_i:E_j) \in S+ \cup S-; p \in Plist(E_i:E_j)} |(E_i, p, E_j)| \quad (6)$$

The reliability of pattern is the average strength of association across each positive seed pair:

$$r(p) = \frac{\sum_{s \in S+} \frac{pmi(p,s)}{max\_pmi}}{|S+|} \quad (7)$$

Here *max_pmi* is used for the normalization purpose, which is the maximum PMI between all patterns and all positive seed pairs.

4.3 Exploiting the Patterns

4.3.1 Patterns Features

One strategy is to directly use the reference patterns as a set of features for classifier learning and testing. To select the most effective patterns for the learner, we rank the patterns according to their scores and then choose the top patterns (first 100 in our study) as the features.

As mentioned, the frequency score is based on the commonality of a pattern to the positive seed pairs. However, if a pattern also occurs frequently for the negative seed pairs, it should be not deemed a good feature as it may lead to many false positive pairs during real resolution. To take this factor into account, we filter the patterns based on their accuracy, which is defined as follows:

$$Accuracy(p) = \frac{|\{s|s \in S+, p \in PList(s)\}|}{|\{s|s \in S+ \cup S-, p \in PList(s)\}|} \quad (8)$$

A pattern with an accuracy below threshold 0.5 is eliminated from the reference pattern set. The remaining patterns are sorted as normal, from which the top 100 patterns are selected as features.

		NWire			NPaper			BNews		
		R	P	F	R	P	F	R	P	F
Normal Features		54.5	80.3	64.9	56.6	76.0	64.9	52.7	75.3	62.0
+ "X such as Y"	proper names	55.1	79.0	64.9	56.8	76.1	65.0	52.6	75.1	61.9
	all types	55.1	78.3	64.7	56.8	74.7	64.4	53.0	74.4	61.9
+ "X and other Y"	proper names	54.7	79.9	64.9	56.4	75.9	64.7	52.6	74.9	61.8
	all types	54.8	79.8	65.0	56.4	75.9	64.7	52.8	73.3	61.4
+ pattern features (frequency)	proper names	58.7	75.8	66.2	57.5	73.9	64.7	54.0	71.1	61.4
	all types	59.7	67.3	63.3	57.4	62.4	59.8	55.9	57.7	56.8
+ pattern features (filtered frequency)	proper names	57.8	79.1	66.8	56.9	75.1	64.7	54.1	72.4	61.9
	all types	58.1	77.4	66.4	56.8	71.2	63.2	55.0	68.1	60.9
+ pattern features (PMI_reliability)	proper names	58.8	76.9	66.6	58.1	73.8	65.0	54.3	72.0	61.9
	all types	59.6	70.4	64.6	58.7	61.6	60.1	56.0	58.8	57.4
+ single reliability feature	proper names	57.4	80.8	**67.1**	56.6	76.2	**65.0**	54.0	74.7	**62.7**
	all types	57.7	76.4	65.7	56.7	75.9	64.9	55.1	69.5	61.5

Table 1: The results of different systems for coreference resolution

Each selected pattern p is used as a single feature, PF_p. For an instance $\boldsymbol{i}\{\text{NP}_i, \text{NP}_j\}$, a list of patterns is generated for (E_i:E_j) in the same way as described in Section 4.1. The value of PF_p for the instance is simply $|(\text{E}_i, \text{p}, \text{E}_j)|$.

The set of pattern features is used together with the other normal features to do the learning and testing. Thus, the actual importance of a pattern in coreference resolution is automatically determined in a supervised learning way.

4.3.2 Semantic Relatedness Feature

Another strategy is to use only one semantic feature which is able to reflect the reliability that a NP pair is related in semantics. Intuitively, a NP pair with strong semantic relatedness should be highly associated with as many reliable patterns as possible. Based on this idea, we define the semantic relatedness feature (SRel) as follows:

$$SRel(\boldsymbol{i}\{NP_i, NP_j\}) = 1000 * \sum_{p \in PList(E_i:E_j)} pmi(p, (E_i : E_j)) * r(p) \quad (9)$$

where pmi(p, (E_i:E_j)) is the pointwise mutual information between pattern p and a NP pair (E_i:E_j), as defined in Eq. 3. $r(p)$ is the reliability score of p (Eq. 7). As a relatedness value is always below 1, we multiple it by 1000 so that the feature value will be of integer type with a range from 0 to 1000. Note that among PList(E_i:E_j), only the reference patterns are involved in the feature computing.

5 Experiments and Discussion

5.1 Experimental setup

In our study we did evaluation on the ACE-2 V1.0 corpus (NIST, 2003), which contains two data set, training and devtest, used for training and testing respectively. Each of these sets is further divided by three domains: newswire (NWire), newspaper (NPaper), and broadcast news (BNews).

An input raw text was preprocessed automatically by a pipeline of NLP components, including sentence boundary detection, POS-tagging, Text Chunking and Named-Entity Recognition. Two different classifiers were learned respectively for resolving pronouns and non-pronouns. As mentioned, the pattern based semantic information was only applied to the non-pronoun resolution. For evaluation, Vilain et al. (1995)'s scoring algorithm was adopted to compute the recall and precision of the whole coreference resolution.

For pattern extraction and feature computing, we used Wikipedia, a web-based free-content encyclopedia, as the text corpus. We collected the English Wikipedia database dump in November 2006 (refer to http://download.wikimedia.org/). After all the hyperlinks and other html tags were removed, the whole pure text contains about 220 Million words.

5.2 Results and Discussion

Table 1 lists the performance of different coreference resolution systems. The first line of the table shows the baseline system that uses only the common features proposed in (Ng and Cardie, 2002). From the table, our baseline system can

NO	Frequency	Frequency (Filtered)	PMI_Reliabilty
1	<#t1> <#t2>	<#t2> | | <#t1> |	<#t1> : <#t2>
2	<#t2> <#t1>	<#t1>) is a <#t2>	<#t2> : <#t1>
3	<#t1> , <#t2>	<#t1>) is an <#t2>	<#t1> . the <#t2>
4	<#t2> , <#t1>	<#t2>) is an <#t1>	<#t2> (<#t1>)
5	<#t1> . <#t2>	<#t2>) is a <#t1>	<#t1> (<#t2>
6	<#t1> and <#t2>	<#t1> or the <#t2>	<#t1> (<#t2>)
7	<#t2> . <#t1>	<#t1> (the <#t2>	<#t1> | | <#t2> |
8	<#t1> . the <#t2>	<#t1> . during the <#t2>	<#t2> | | <#t1> |
9	<#t2> and <#t1>	<#t1> | <#t2>	<#t2> , the <#t1>
10	<#t1> , the <#t2>	<#t1> , an <#t2>	<#t1> , the <#t2>
11	<#t2> . the <#t1>	<#t1>) was a <#t2>	<#t2> (<#t1>
12	<#t2> , the <#t1>	<#t1> in the <#t2> -	<#t1> , <#t2>
13	<#t2> <#t1> ,	<#t1> - <#t2>	<#t1> and the <#t2>
14	<#t1> <#t2> ,	<#t1>) was an <#t2>	<#t1> . <#t2>
15	<#t1> : <#t2>	<#t1> , many <#t2>	<#t1>) is a <#t2>
16	<#t1> <#t2> .	<#t2>) was a <#t1>	<#t1> during the <#t2>
17	<#t2> <#t1> .	<#t1> (<#t2> .	<#t1> <#t2> .
18	<#t1> (<#t2>)	<#t2> | <#t1>	<#t1>) is an <#t2>
19	<#t1> and the <#t2>	<#t1> , not the <#t2>	<#t2> in <#t1> .
20	<#t2> (<#t1>)	<#t2> , many <#t1>	<#t2> , <#t1>
...	...	...	...

Table 2: Top patterns chosen under different scoring schemes

achieve a good precision (above 75%-80%) with a recall around 50%-60%. The overall F-measure for NWire, NPaper and BNews is 64.9%, 64.9% and 62.0% respectively. The results are comparable to those reported in (Ng, 2005) which uses similar features and gets an F-measure of about 62% for the same data set.

The rest lines of Table 1 are for the systems using the pattern based information. In all the systems, we examine the utility of the semantic information in resolving different types of NP Pairs: (1) NP Pairs containing proper names (i.e., Name:Name or Name:Definites), and (2) NP Pairs of all types.

In Table 1 (Line 2-5), we also list the results of incorporating two commonly used patterns, "X(s) such as Y" and "X and other Y(s)". We can find that neither of the manually designed patterns has significant impact on the resolution performance. For all the domains, the manual patterns just achieve slight improvement in recall (below 0.6%), indicating that coverage of the patterns is not broad enough.

5.2.1 Pattern Features

In Section 4.3.1 we propose a strategy that directly uses the patterns as features. Table 2 lists the top patterns that are sorted based on *frequency*, *filtered frequency (by accuracy)*, and *PMI_reliability*, on the NWire domain for illustration.

From the table, evaluated only based on *frequency*, the top patterns are those that indicate the appositive structure like "*X*, *an/a/the Y*". However, if filtered by *accuracy*, patterns of such a kind will be removed. Instead, the top patterns with both high frequency and high accuracy are those for the copula structure, like "*X is/was/are Y*". Sorted by PMI reliability, patterns for the above two structures can be seen in the top of the list. These results are consistent with the findings in (Cimiano and Staab, 2004) that the appositive and copula structures are indicative to find the *is-a* relation. Also, the two commonly used patterns "X(s) such as Y" and "X and other Y(s)" were found in the feature lists (not shown in the table). Their importance for coreference resolution will be determined automatically by the learning algorithm.

An interesting pattern seen in the lists is "$X \mid\mid Y \mid$", which represents the cases when Y and X appear in the same of line of a table in Wikipedia. For example, the following text
"*American* || *United States* | *Washington D.C.* | ..."
is found in the table "list of empires". Thus the pair "American:United States", which is deemed coreferential in ACE, can be identified by the pattern.

The sixth till the eleventh lines of Table 1 list the results of the system with pattern features. From the table, adding the pattern features brings the improvement of the recall against the baseline. Take the system based on *filtered frequency* as an example. We can observe that the recall increases by up to 3.3% (for NWire). However, we see the precision drops (up to 1.2% for NWire) at the same time. Overall the system achieves an F-measure better than the baseline in NWire (1.9%), while equal ($\pm$0.2%) in NPaper and BNews.

Among the three ranking schemes, simply using *frequency* leads to the lowest precision. By contrast, using *filtered frequency* yields the highest precision with nevertheless the lowest recall. It is reasonable since the low_accuracy features prone to false posi-

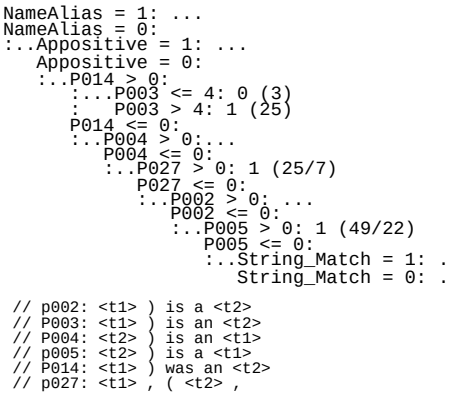


Figure 1: The decision tree (NWire domain) for the system using pattern features (filtered frequency)
(feature *String_Match* records whether the string of anaphor NP_j matches that of a candidate antecedent NP_i)

tive NP pairs are eliminated, at the price of recall. Using *PMI_Reliability* can achieve the highest recall with a medium level of precision. However, we do not find significant difference in the overall F-measure for all these three schemes. This should be due to the fact that the pattern features need to be further chosen by the learning algorithm, and only those patterns deemed effective by the learner will really matter in the real resolution.

From the table, the pattern features only work well for NP pairs containing proper names. Applied on all types of NP pairs, the pattern features further boost the recall of the systems, but in the meanwhile degrade the precision significantly. The F-measure of the systems is even worse than that of the baseline. Our error analysis shows that a non-anaphor is often wrongly resolved to a false antecedent once the two NPs happen to satisfy a pattern feature, which affects precision largely (as an evidence, the decrease of precision is less significant when using *filtered_frequency* than using *frequency*). Still, these results suggest that we just apply the pattern based semantic information in resolving proper names which, in fact, is more compelling as the semantic information of common nouns could be more easily retrieved from WordNet.

We also notice that the patterned based semantic information seems more effective in the NWire domain than the other two. Especially for NPaper, the improvement in F-measure is less than 0.1% for all the systems tested. The error analysis indicates it may be because (1) there are less NP pairs in NPaper than in NWire that require the external semantic knowledge for resolution; and (2) For many NP pairs that require the semantic knowledge, no co-occurrence can be found in the Wikipedia corpus. To address this problem, we could resort to the Web which contains a larger volume of texts and thus could lead to more informative patterns. We would like to explore this issue in our future work.

In Figure 1, we plot the decision tree learned with the pattern features for non-pronoun resolution (NWire domain, *filtered_frequency*), which visually illustrates which features are useful in the reference determination. We can find the pattern features occur in the top of the decision tree, among the features for *name_alias*, *apposition* and *string-matching* that are crucial for coreference resolution as reported in previous work (Soon et al., 2001). Most of the pattern features deemed important by the learner are for the copula structure.

5.2.2 Single Semantic Relatedness Feature

Section 4.3.2 presents another strategy to exploit the patterns, which uses a single feature to reflect the semantic relatedness between NP pairs. The last two lines of Table 1 list the results of such a system.

Observed from the table, the system with the single semantic relatedness feature beats those with other solutions. Compared with the baseline, the system can get improvement in recall (up to 2.9% as in NWire), with a similar or even higher precision. The overall F-measure it produces is 67.1%, 65.0% and 62.7%, better than the baseline in all the domains. Especially in the NWire domain, we can see the significant (t-test, $p \leq 0.05$) improvement of 2.1% in F-measure. When applied on All-Type NP pairs, the degrade of performance is less significant as using pattern features. The resulting performance is better than the baseline or equal. Compared with the systems using the pattern features, it can still achieve a higher precision and F-measure (with a little loss in recall) .

There are several reasons why the single semantic relatedness feature (*SRel*) can perform better than the set of pattern features. Firstly, the feature value of *SRel* takes into consideration the information of all the patterns, instead of only the selected patterns. Secondly, since the *SRel* feature is computed based on all the patterns, it reduces the risk of false posi-

```
NameAlias = 1: ...
NameAlias = 0:
:..Appositive = 1: ...
   Appositive = 0:
   :...SRel > 28:
      :...SRel > 47: ...
      :   SRel <= 47: ...
      SRel <= 28:
      :...String_Match = 1: ...
          String_Match = 0: ...
```

Figure 2: The decision tree (Nwire) for the system using the single semantic relatedness feature

tive when a NP pair happens to satisfy one or several pattern features. Lastly, from the point of view of machine learning, using only one semantic feature, instead of hundreds of pattern features, can avoid overfitting and thus benefit the classifier learning.

In Figure 2, we also show the decision tree learned with the semantic relatedness feature. We observe that the decision tree is simpler than that with pattern features as depicted in Figure 1. After feature *name-alias* and *apposite*, the classifier checks different ranges of the *SRel* value and make different resolution decision accordingly. This figure further illustrates the importance of the semantic feature.

6 Conclusions

In this paper we present a pattern based approach to coreference resolution. Different from the previous work which utilizes manually designed patterns, our approach can automatically discover the patterns effective for the coreference resolution task. In our study, we explore how to acquire and evaluate patterns, and investigate how to exploit the patterns to mine semantic relatedness information for coreference resolution. The evaluation on ACE data set shows that the patterned based features, when applied on NP pairs containing proper names, can effectively help the performance of coreference resolution in the recall (up to 4.3%) and the overall F-measure (up to 2.1%). The results also indicate that using the single semantic relatedness feature has more advantages than using a set of pattern features.

For future work, we intend to investigate our approach in more difficult tasks like the bridging anaphora resolution, in which the semantic relations involved are more complicated. Also, we would like to explore the approach in technical (e.g., biomedical) domains, where jargons are frequently seen and the need for external knowledge is more compelling.

Acknowledgements This research is supported by a Specific Targeted Research Project (STREP) of the European Union's 6th Framework Programme within IST call 4, Bootstrapping Of Ontologies and Terminologies STrategic REsearch Project (BOOTStrep).

References

D. Bean and E. Riloff. 2004. Unsupervised learning of contextual role knowledge for coreference resolution. In *Proceedings of NAACL*, pages 297–304.

P. Cimiano and S. Staab. 2004. Learning by googling. *SIGKDD Explorations Newsletter*, 6(2):24–33.

T. Cover and J. Thomas. 1991. *Elements of Information Theory*. Hohn Wiley & Sons.

N. Garera and D. Yarowsky. 2006. Resolving and generating definite anaphora by modeling hypernymy using unlabeled corpora. In *Proceedings of CoNLL* , pages 37–44.

S. Harabagiu, R. Bunescu, and S. Maiorano. 2001. Text knowledge mining for coreference resolution. In *Proceedings of NAACL*, pages 55–62.

M. Hearst. 1998. Automated discovery of wordnet relations. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database and Some of its Applications*. MIT Press, Cambridge, MA.

K. Markert, M. Nissim, and N. Modjeska. 2003. Using the web for nominal anaphora resolution. In *Proceedings of the EACL workshop on Computational Treatment of Anaphora*, pages 39–46.

N. Modjeska, K. Markert, and M. Nissim. 2003. Using the web in machine learning for other-anaphora resolution. In *Proceedings of EMNLP*, pages 176–183.

V. Ng and C. Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of ACL*, pages 104–111, Philadelphia.

V. Ng. 2005. Machine learning for coreference resolution: From local classification to global ranking. In *Proceedings of ACL*, pages 157–164.

P. Pantel and M. Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of ACL*, pages 113–1200.

M. Poesio, R. Mehta, A. Maroudas, and J. Hitzeman. 2004. Learning to resolve bridging references. In *Proceedings of ACL*, pages 143–150.

S. Ponzetto and M. Strube. 2006. Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In *Proceedings of NAACL*, pages 192–199.

W. Soon, H. Ng, and D. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

R. Vieira and M. Poesio. 2000. An empirically based system for processing definite descriptions. *Computational Linguistics*, 27(4):539–592.

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the Sixth Message understanding Conference (MUC-6)*, pages 45–52, San Francisco, CA. Morgan Kaufmann Publishers.

Semantic Class Induction and Coreference Resolution

Vincent Ng
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
vince@hlt.utdallas.edu

Abstract

This paper examines whether a learning-based coreference resolver can be improved using semantic class knowledge that is automatically acquired from a version of the Penn Treebank in which the noun phrases are labeled with their semantic classes. Experiments on the ACE test data show that a resolver that employs such induced semantic class knowledge yields a statistically significant improvement of 2% in F-measure over one that exploits heuristically computed semantic class knowledge. In addition, the induced knowledge improves the accuracy of common noun resolution by 2-6%.

1 Introduction

In the past decade, *knowledge-lean* approaches have significantly influenced research in noun phrase (NP) coreference resolution — the problem of determining which NPs refer to the same real-world entity in a document. In knowledge-lean approaches, coreference resolvers employ only morpho-syntactic cues as knowledge sources in the resolution process (e.g., Mitkov (1998), Tetreault (2001)). While these approaches have been reasonably successful (see Mitkov (2002)), Kehler et al. (2004) speculate that deeper linguistic knowledge needs to be made available to resolvers in order to reach the next level of performance. In fact, semantics plays a crucially important role in the resolution of common NPs, allowing us to identify the coreference relation between two lexically dissimilar common nouns (e.g., *talks* and *negotiations*) and to eliminate *George W. Bush* from the list of candidate antecedents of *the city*, for instance. As a result, researchers have re-adopted the once-popular *knowledge-rich* approach, investigating a variety of semantic knowledge sources for common noun resolution, such as the *semantic relations* between two NPs (e.g., Ji et al. (2005)), their *semantic similarity* as computed using WordNet (e.g., Poesio et al. (2004)) or Wikipedia (Ponzetto and Strube, 2006), and the *contextual role* played by an NP (see Bean and Riloff (2004)).

Another type of semantic knowledge that has been employed by coreference resolvers is the *semantic class* (SC) of an NP, which can be used to disallow coreference between semantically incompatible NPs. However, learning-based resolvers have not been able to benefit from having an SC agreement feature, presumably because the method used to compute the SC of an NP is too simplistic: while the SC of a proper name is computed fairly accurately using a named entity (NE) recognizer, many resolvers simply assign to a common noun the first (i.e., most frequent) WordNet sense as its SC (e.g., Soon et al. (2001), Markert and Nissim (2005)). It is not easy to measure the accuracy of this heuristic, but the fact that the SC agreement feature is not used by Soon et al.'s decision tree coreference classifier seems to suggest that the SC values of the NPs are not computed accurately by this first-sense heuristic.

Motivated in part by this observation, we examine whether automatically induced semantic class knowledge can improve the performance of a learning-based coreference resolver, reporting evaluation results on the commonly-used ACE corefer-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 536–543,
Prague, Czech Republic, June 2007.

ence corpus. Our investigation proceeds as follows.

Train a classifier for labeling the SC of an NP. In ACE, we are primarily concerned with classifying an NP as belonging to one of the ACE semantic classes. For instance, part of the ACE Phase 2 evaluation involves classifying an NP as PERSON, ORGANIZATION, GPE (a geographical-political region), FACILITY, LOCATION, or OTHERS. We adopt a corpus-based approach to SC determination, recasting the problem as a six-class classification task.

Derive two knowledge sources for coreference resolution from the induced SCs. The first knowledge source (KS) is *semantic class agreement* (SCA). Following Soon et al. (2001), we represent SCA as a binary value that indicates whether the induced SCs of the two NPs involved are the same or not. The second KS is *mention*, which is represented as a binary value that indicates whether an NP belongs to one of the five ACE SCs mentioned above. Hence, the *mention* value of an NP can be readily derived from its induced SC: the value is NO if its SC is OTHERS, and YES otherwise. This KS could be useful for ACE coreference, since ACE is concerned with resolving only NPs that are mentions.

Incorporate the two knowledge sources in a coreference resolver. Next, we investigate whether these two KSs can improve a learning-based baseline resolver that employs a fairly standard feature set. Since (1) the two KSs can each be represented in the resolver as a *constraint* (for filtering non-mentions or disallowing coreference between semantically incompatible NPs) or as a *feature*, and (2) they can be applied to the resolver in *isolation* or in *combination*, we have eight ways of incorporating these KSs into the baseline resolver.

In our experiments on the ACE Phase 2 coreference corpus, we found that (1) our SC induction method yields a significant improvement of 2% in accuracy over Soon et al.'s first-sense heuristic method as described above; (2) the coreference resolver that incorporates our induced SC knowledge by means of the two KSs mentioned above yields a significant improvement of 2% in F-measure over the resolver that exploits the SC knowledge computed by Soon et al.'s method; (3) the *mention* KS, when used in the baseline resolver as a *constraint*, improves the resolver by approximately 5-7% in F-measure; and (4) *SCA*, when employed as a *feature* by the baseline resolver, improves the accuracy of common noun resolution by about 5-8%.

2 Related Work

Mention detection. Many ACE participants have also adopted a corpus-based approach to SC determination that is investigated as part of the *mention detection* (MD) task (e.g., Florian et al. (2006)). Briefly, the goal of MD is to identify the boundary of a mention, its mention type (e.g., pronoun, name), and its semantic type (e.g., person, location). Unlike them, (1) we do not perform the full MD task, as our goal is to investigate the role of SC knowledge in coreference resolution; and (2) we do not use the ACE training data for acquiring our SC classifier; instead, we use the *BBN Entity Type Corpus* (Weischedel and Brunstein, 2005), which consists of all the Penn Treebank Wall Street Journal articles with the ACE mentions manually identified and annotated with their SCs. This provides us with a training set that is approximately five times bigger than that of ACE. More importantly, the ACE participants do not evaluate the role of *induced* SC knowledge in coreference resolution: many of them evaluate coreference performance on perfect mentions (e.g., Luo et al. (2004)); and for those that do report performance on automatically extracted mentions, they do not explain whether or how the induced SC information is used in their coreference algorithms.

Joint probabilistic models of coreference. Recently, there has been a surge of interest in improving coreference resolution by jointly modeling coreference with a related task such as MD (e.g., Daumé and Marcu (2005)). However, joint models typically need to be trained on data that is *simultaneously* annotated with information required by all of the underlying models. For instance, Daumé and Marcu's model assumes as input a corpus annotated with both MD and coreference information. On the other hand, we tackle coreference and SC induction separately (rather than jointly), since we train our SC determination model on the BBN Entity Type Corpus, where coreference information is absent.

3 Semantic Class Induction

This section describes how we train and evaluate a classifier for determining the SC of an NP.

3.1 Training the Classifier

Training corpus. As mentioned before, we use the BBN Entity Type Corpus for training the SC classifier. This corpus was originally developed to support the ACE and AQUAINT programs and consists of annotations of 12 named entity types and nine nominal entity types. Nevertheless, we will only make use of the annotations of the five ACE semantic types that are present in our ACE Phase 2 coreference corpus, namely, PERSON, ORGANIZATION, GPE, FACILITY, and LOCATION.

Training instance creation. We create one training instance for each proper or common NP (extracted using an NP chunker and an NE recognizer) in each training text. Each instance is represented by a set of lexical, syntactic, and semantic features, as described below. If the NP under consideration is annotated as one of the five ACE SCs in the corpus, then the classification of the associated training instance is simply the ACE SC value of the NP. Otherwise, the instance is labeled as OTHERS. This results in 310063 instances in the training set.

Features. We represent the training instance for a noun phrase, NP_i, using seven types of features:

(1) WORD: For each word w in NP_i, we create a WORD feature whose value is equal to w. No features are created from stopwords, however.

(2) SUBJ_VERB: If NP_i is involved in a subject-verb relation, we create a SUBJ_VERB feature whose value is the verb participating in the relation. We use Lin's (1998b) MINIPAR dependency parser to extract grammatical relations. Our motivation here is to coarsely model subcategorization.

(3) VERB_OBJ: A VERB_OBJ feature is created in a similar fashion as SUBJ_VERB if NP_i participates in a verb-object relation. Again, this represents our attempt to coarsely model subcategorization.

(4) NE: We use BBN's IdentiFinder (Bikel et al., 1999), a MUC-style NE recognizer to determine the NE type of NP_i. If NP_i is determined to be a PERSON or ORGANIZATION, we create an NE feature whose value is simply its MUC NE type. However, if NP_i is determined to be a LOCATION, we create a feature with value GPE (because most of the MUC LOCATION NEs are ACE GPE NEs). Otherwise, no NE feature will be created (because we are not interested in the other MUC NE types).

ACE SC	Keywords
PERSON	person
ORGANIZATION	social group
FACILITY	establishment, construction, building, facility, workplace
GPE	country, province, government, town, city, administration, society, island, community
LOCATION	dry land, region, landmass, body of water, geographical area, geological formation

Table 1: List of keywords used in WordNet search for generating WN_CLASS features.

(5) WN_CLASS: For each keyword w shown in the right column of Table 1, we determine whether the head noun of NP_i is a hyponym of w in WordNet, using only the first WordNet sense of NP_i.[1] If so, we create a WN_CLASS feature with w as its value. These keywords are potentially useful features because some of them are subclasses of the ACE SCs shown in the left column of Table 1, while others appear to be correlated with these ACE SCs.[2]

(6) INDUCED_CLASS: Since the first-sense heuristic used in the previous feature may not be accurate in capturing the SC of an NP, we employ a corpus-based method for inducing SCs that is motivated by research in lexical semantics (e.g., Hearst (1992)). Given a large, unannotated corpus[3], we use IdentiFinder to label each NE with its NE type and MINIPAR to extract all the appositive relations. An example extraction would be <*Eastern Airlines*, *the carrier*>, where the first entry is a proper noun labeled with either one of the seven MUC-style NE types[4] or OTHERS[5] and the second entry is a common noun. We then infer the SC of a common noun as follows: (1) we compute the probability that the common noun co-occurs with each of the eight NE types[6] based on the extracted appositive relations, and (2) if the most likely NE type has a co-occurrence probability above a certain threshold (we set it to 0.7), we create a INDUCED_CLASS fea-

[1] This is motivated by Lin's (1998c) observation that a coreference resolver that employs only the first WordNet sense performs slightly better than one that employs more than one sense.

[2] The keywords are obtained via our experimentation with WordNet and the ACE SCs of the NPs in the ACE training data.

[3] We used (1) the BLLIP corpus (30M words), which consists of WSJ articles from 1987 to 1989, and (2) the Reuters Corpus (3.7GB data), which has 806,791 Reuters articles.

[4] Person, organization, location, date, time, money, percent.

[5] This indicates the proper noun is not a MUC NE.

[6] For simplicity, OTHERS is viewed as an NE type here.

ture for NP_i whose value is the most likely NE type.
(7) NEIGHBOR: Research in lexical semantics suggests that the SC of an NP can be inferred from its distributionally similar NPs (see Lin (1998a)). Motivated by this observation, we create for each of NP_i's ten most semantically similar NPs a NEIGHBOR feature whose value is the surface string of the NP. To determine the ten nearest neighbors, we use the semantic similarity values provided by Lin's dependency-based thesaurus, which is constructed using a distributional approach combined with an information-theoretic definition of similarity.

Learning algorithms. We experiment with four learners commonly employed in language learning:
Decision List (DL): We use the DL learner as described in Collins and Singer (1999), motivated by its success in the related tasks of word sense disambiguation (Yarowsky, 1995) and NE classification (Collins and Singer, 1999). We apply add-one smoothing to smooth the class posteriors.
1-Nearest Neighbor (1-NN): We use the 1-NN classifier as implemented in TiMBL (Daelemans et al., 2004), employing *dot product* as the similarity function (which defines similarity as the number of common feature-value pairs between two instances). All other parameters are set to their default values.
Maximum Entropy (ME): We employ Lin's ME implementation[7], using a Gaussian prior for smoothing and running the algorithm until convergence.
Naive Bayes (NB): We use an in-house implementation of NB, using add-one smoothing to smooth the class priors and the class-conditional probabilities.

In addition, we train an SVM classifier for SC determination by combining the output of five classification methods: DL, 1-NN, ME, NB, and Soon et al.'s method as described in the introduction,[8] with the goal of examining whether SC classification accuracy can be improved by combining the output of individual classifiers in a supervised manner. Specifically, we (1) use 80% of the instances generated from the BBN Entity Type Corpus to train the four classifiers; (2) apply the four classifiers and Soon et al.'s method to independently make predictions for the remaining 20% of the instances; and (3) train an SVM classifier (using the LIBSVM package (Chang and Lin, 2001)) on these 20% of the instances, where each instance, i, is represented by a set of 31 binary features. More specifically, let $L_i = \{l_{i1}, l_{i2}, l_{i3}, l_{i4}, l_{i5}\}$ be the set of predictions that we obtained for i in step (2). To represent i, we generate one feature from each non-empty subset of L_i.

[7] See http://www.cs.ualberta.ca/∼lindek/downloads.htm

[8] In our implementation of Soon's method, we label an instance as OTHERS if no NE or WN_CLASS feature is generated; otherwise its label is the value of the NE feature or the ACE SC that has the WN_CLASS features as its keywords (see Table 1).

	PER	ORG	GPE	FAC	LOC	OTH
Training	19.8	9.6	11.4	1.6	1.2	56.3
Test	19.5	9.0	9.6	1.8	1.1	59.0

Table 2: Distribution of SCs in the ACE corpus.

3.2 Evaluating the Classifiers

For evaluation, we use the ACE Phase 2 coreference corpus, which comprises 422 training texts and 97 test texts. Each text has its mentions annotated with their ACE SCs. We create our test instances from the ACE texts in the same way as the training instances described in Section 3.1. Table 2 shows the percentages of instances corresponding to each SC.

Table 3 shows the accuracy of each classifier (see row 1) for the ACE training set (54641 NPs, with 16414 proper NPs and 38227 common NPs) and the ACE test set (13444 NPs, with 3713 proper NPs and 9731 common NPs), as well as their performance on the proper NPs (row 2) and the common NPs (row 3). We employ as our baseline system the Soon et al. method (see Footnote 8), whose accuracy is shown under the Soon column. As we can see, DL, 1-NN, and SVM show a statistically significant improvement over the baseline for both data sets, whereas ME and NB perform significantly worse.[9] Additional experiments are needed to determine the reason for ME and NB's poor performance.

In an attempt to gain additional insight into the performance contribution of each type of features, we conduct feature ablation experiments using the DL classifier (DL is chosen simply because it is the best performer on the ACE training set). Results are shown in Table 4, where each row shows the accuracy of the DL trained on all types of features except for the one shown in that row (All), as well as accuracies on the proper NPs (PN) and the common NPs (CN). For easy reference, the accuracy of the DL

[9] We use Noreen's (1989) Approximate Randomization test for significance testing, with p set to .05 unless otherwise stated.

		Training Set						Test Set					
		Soon	DL	1-NN	ME	NB	SVM	Soon	DL	1-NN	ME	NB	SVM
1	Overall	83.1	**85.0**	84.0	54.5	71.3	84.2	81.1	82.9	83.1	53.0	70.3	**83.3**
2	Proper NPs	83.1	**84.1**	81.0	54.2	65.5	82.2	79.6	**82.0**	79.8	55.8	64.4	80.4
3	Common NPs	83.1	**85.4**	85.2	54.6	73.8	85.1	81.6	83.3	84.3	51.9	72.6	**84.4**

Table 3: SC classification accuracies of different methods for the ACE training set and test set.

	Training Set			Test Set		
Feature Type	PN	CN	All	PN	CN	All
All features	84.1	85.4	85.0	82.0	83.3	82.9
- WORD	84.2	85.4	85.0	82.0	83.1	82.8
- SUBJ_VERB	84.1	85.4	85.0	82.0	83.3	82.9
- VERB_OBJ	84.1	85.4	85.0	82.0	83.3	82.9
- NE	72.9	85.3	81.6	74.1	83.2	80.7
- WN_CLASS	84.1	85.9	85.3	81.9	84.1	83.5
- INDUCED_C	84.0	85.6	85.1	82.0	83.6	83.2
- NEIGHBOR	82.8	84.9	84.3	80.2	82.9	82.1

Table 4: Results for feature ablation experiments.

	Training Set			Test Set		
Feature Type	PN	CN	All	PN	CN	All
WORD	64.0	83.9	77.9	66.5	82.4	78.0
SUBJ_VERB	24.0	70.2	56.3	28.8	70.5	59.0
VERB_OBJ	24.0	70.2	56.3	28.8	70.5	59.0
NE	81.1	72.1	74.8	78.4	71.4	73.3
WN_CLASS	25.6	78.8	62.8	30.4	78.9	65.5
INDUCED_C	25.8	81.1	64.5	30.0	80.3	66.3
NEIGHBOR	67.7	85.8	80.4	68.0	84.4	79.8

Table 5: Accuracies of single-feature classifiers.

classifier trained on all types of features is shown in row 1 of the table. As we can see, accuracy drops significantly with the removal of NE and NEIGHBOR. As expected, removing NE precipitates a large drop in proper NP accuracy; somewhat surprisingly, removing NEIGHBOR also causes proper NP accuracy to drop significantly. To our knowledge, there are no prior results on using distributionally similar neighbors as features for supervised SC induction.

Note, however, that these results do not imply that the remaining feature types are not useful for SC classification; they simply suggest, for instance, that WORD is not important in the presence of other feature types. To get a better idea of the utility of each feature type, we conduct another experiment in which we train seven classifiers, each of which employs exactly one type of features. The accuracies of these classifiers are shown in Table 5. As we can see, NEIGHBOR has the largest contribution. This again demonstrates the effectiveness of a distributional approach to semantic similarity. Its superior performance to WORD, the second largest contributor, could be attributed to its ability to combat data sparseness. The NE feature, as expected, is crucial to the classification of proper NPs.

4 Application to Coreference Resolution

We can now derive from the induced SC information two KSs — *semantic class agreement* and *mention* — and incorporate them into our learning-based coreference resolver in eight different ways, as described in the introduction. This section examines whether our coreference resolver can benefit from any of the eight ways of incorporating these KSs.

4.1 Experimental Setup

As in SC induction, we use the ACE Phase 2 coreference corpus for evaluation purposes, acquiring the coreference classifiers on the 422 training texts and evaluating their output on the 97 test texts. We report performance in terms of two metrics: (1) the *F-measure* score as computed by the commonly-used MUC scorer (Vilain et al., 1995), and (2) the *accuracy* on the anaphoric references, computed as the fraction of anaphoric references correctly resolved. Following Ponzetto and Strube (2006), we consider an anaphoric reference, NP_i, correctly resolved if NP_i and its closest antecedent are in the same coreference chain in the resulting partition. In all of our experiments, we use NPs automatically extracted by an in-house NP chunker and IdentiFinder.

4.2 The Baseline Coreference System

Our baseline coreference system uses the C4.5 decision tree learner (Quinlan, 1993) to acquire a classifier on the training texts for determining whether two NPs are coreferent. Following previous work (e.g., Soon et al. (2001) and Ponzetto and Strube (2006)), we generate training instances as follows: a positive instance is created for each anaphoric NP, NP_j, and its closest antecedent, NP_i; and a negative instance is created for NP_j paired with each of the intervening NPs, NP_{i+1}, NP_{i+2}, ..., NP_{j-1}. Each instance is represented by 33 lexical, grammatical, semantic, and

positional features that have been employed by high-performing resolvers such as Ng and Cardie (2002) and Yang et al. (2003), as described below.

Lexical features. Nine features allow different types of string matching operations to be performed on the given pair of NPs, NP_x and NP_y[10], including (1) exact string match for pronouns, proper nouns, and non-pronominal NPs (both before and after determiners are removed); (2) substring match for proper nouns and non-pronominal NPs; and (3) head noun match. In addition, one feature tests whether all the words that appear in one NP also appear in the other NP. Finally, a nationality matching feature is used to match, for instance, *British* with *Britain*.

Grammatical features. 22 features test the grammatical properties of one or both of the NPs. These include ten features that test whether each of the two NPs is a pronoun, a definite NP, an indefinite NP, a nested NP, and a clausal subject. A similar set of five features is used to test whether both NPs are pronouns, definite NPs, nested NPs, proper nouns, and clausal subjects. In addition, five features determine whether the two NPs are compatible with respect to gender, number, animacy, and grammatical role. Furthermore, two features test whether the two NPs are in apposition or participate in a predicate nominal construction (i.e., the IS-A relation).

Semantic features. Motivated by Soon et al. (2001), we have a semantic feature that tests whether one NP is a name alias or acronym of the other.

Positional feature. We have a feature that computes the distance between the two NPs in sentences.

After training, the decision tree classifier is used to select an antecedent for each NP in a test text. Following Soon et al. (2001), we select as the antecedent of each NP, NP_j, the *closest* preceding NP that is classified as coreferent with NP_j. If no such NP exists, no antecedent is selected for NP_j.

Row 1 of Table 6 and Table 7 shows the results of the baseline system in terms of F-measure (F) and accuracy in resolving 4599 anaphoric references (All), respectively. For further analysis, we also report the corresponding recall (R) and precision (P) in Table 6, as well as the accuracies of the system in resolving 1769 pronouns (PRO), 1675 proper NPs (PN), and 1155 common NPs (CN) in Table 7. As we can see, the baseline achieves an F-measure of 57.0 and a resolution accuracy of 48.4.

To get a better sense of how strong our baseline is, we re-implement the Soon et al. (2001) coreference resolver. This simply amounts to replacing the 33 features in the baseline resolver with the 12 features employed by Soon et al.'s system. Results of our Duplicated Soon et al. system are shown in row 2 of Tables 6 and 7. In comparison to our baseline, the Duplicated Soon et al. system performs worse according to both metrics, and although the drop in F-measure seems moderate, the performance difference is in fact highly significant (p=0.002).[11]

4.3 Coreference with Induced SC Knowledge

Recall from the introduction that our investigation of the role of induced SC knowledge in learning-based coreference resolution proceeds in three steps:

Label the SC of each NP in each ACE document. If a noun phrase, NP_i, is a proper or common NP, then its SC value is determined using an SC classifier that we acquired in Section 3. On the other hand, if NP_i is a pronoun, then we will be conservative and posit its SC value as UNCONSTRAINED (i.e., it is semantically compatible with all other NPs).[12]

Derive two KSs from the induced SCs. Recall that our first KS, *Mention*, is defined on an NP; its value is YES if the induced SC of the NP is not OTHERS, and NO otherwise. On the other hand, our second KS, *SCA*, is defined on a pair of NPs; its value is YES if the two NPs have the same induced SC that is not OTHERS, and NO otherwise.

Incorporate the two KSs into the baseline resolver. Recall that there are eight ways of incorporating these two KSs into our resolver: they can each be represented as a *constraint* or as a *feature*, and they can be applied to the resolver in *isolation* and in *combination*. Constraints are applied during the antecedent selection step. Specifically, when employed as a constraint, the *Mention* KS disallows coreference between two NPs if at least one of them has a *Mention* value of NO, whereas the *SCA* KS disallows coreference if the *SCA* value of the two NPs involved is NO. When encoded as a feature for the resolver, the *Mention* feature for an NP pair has the

[10] We assume that NP_x precedes NP_y in the associated text.

[11] Again, we use Approximate Randomization with p=.05.

[12] The only exception is pronouns whose SC value can be easily determined to be PERSON (e.g., *he*, *him*, *his*, *himself*).

	System Variation	R	P	F	R	P	F	R	P	F	R	P	F
1	Baseline system	60.9	53.6	57.0	–	–	–	–	–	–	–	–	–
2	Duplicated Soon et al.	56.1	54.4	55.3	–	–	–	–	–	–	–	–	–
	Add to the Baseline	Soon's SC Method			Decision List			SVM			Perfect Information		
3	Mention(C) only	56.9	69.7	**62.6**	59.5	70.6	**64.6**	59.5	70.7	**64.6**	61.2	83.1	70.5
4	Mention(F) only	60.9	54.0	57.2	61.2	52.9	56.7	60.9	53.6	57.0	62.3	33.7	43.8
5	SCA(C) only	56.4	70.0	62.5	57.7	71.2	63.7	58.9	70.7	64.3	61.3	86.1	71.6
6	SCA(F) only	62.0	52.8	57.0	62.5	53.5	57.6	63.0	53.3	57.7	71.1	33.0	45.1
7	Mention(C) + SCA(C)	56.4	70.0	62.5	57.7	71.2	63.7	58.9	70.8	64.3	61.3	86.1	71.6
8	Mention(C) + SCA(F)	58.2	66.4	62.0	60.9	66.8	63.7	61.4	66.5	63.8	71.1	76.7	**73.8**
9	Mention(F) + SCA(C)	56.4	69.8	62.4	57.7	71.3	63.8	58.9	70.6	64.3	62.7	85.3	72.3
10	Mention(F) + SCA(F)	62.0	52.7	57.0	62.6	52.8	57.3	63.2	52.6	57.4	71.8	30.3	42.6

Table 6: Coreference results obtained via the MUC scoring program for the ACE test set.

	System Variation	PRO	PN	CN	All	PRO	PN	CN	All	PRO	PN	CN	All
1	Baseline system	59.2	54.8	22.5	48.4	–	–	–	–	–	–	–	–
2	Duplicated Soon et al.	53.4	45.7	16.9	41.4	–	–	–	–	–	–	–	–
	Add to the Baseline	Soon's SC Method				Decision List				SVM			
3	Mention(C) only	58.5	51.3	16.5	45.3	59.1	54.1	20.2	47.5	59.1	53.9	20.6	47.5
4	Mention(F) only	59.2	55.0	22.5	48.5	59.2	56.1	22.4	48.8	59.4	55.2	22.6	48.6
5	SCA(C) only	58.1	50.1	16.4	44.7	58.1	51.8	17.1	45.5	58.5	52.0	19.6	46.3
6	SCA(F) only	59.2	54.9	27.8	**49.7**	60.4	56.7	30.1	**51.5**	60.8	56.4	29.4	51.3
7	Mention(C) + SCA(C)	58.1	50.1	16.4	44.7	58.1	51.8	17.1	45.5	58.5	51.9	19.5	46.3
8	Mention(C) + SCA(F)	58.9	52.0	22.3	47.2	60.2	55.9	28.1	50.6	60.7	55.3	27.4	50.4
9	Mention(F) + SCA(C)	58.1	50.3	16.3	44.8	58.1	52.4	16.7	45.6	58.6	52.4	19.7	46.6
10	Mention(F) + SCA(F)	59.2	55.0	27.6	**49.7**	60.4	56.8	30.1	**51.5**	60.8	56.5	29.5	**51.4**

Table 7: Resolution accuracies for the ACE test set.

value YES if and only if the *Mention* value for both NPs is YES, whereas the *SCA* feature for an NP pair has its value taken from the *SCA* KS.

Now, we can evaluate the impact of the two KSs on the performance of our baseline resolver. Specifically, rows 3-6 of Tables 6 and 7 show the F-measure and the resolution accuracy, respectively, when exactly one of the two KSs is employed by the baseline as either a constraint (C) or a feature (F), and rows 7-10 of the two tables show the results when both KSs are applied to the baseline. Furthermore, each row of Table 6 contains four sets of results, each of which corresponds to a different method for determining the SC value of an NP. For instance, the first set is obtained by using Soon et al.'s method as described in Footnote 8 to compute SC values, serving as sort of a baseline for our results using induced SC values. The second and third sets are obtained based on the SC values computed by the DL and the SVM classifier, respectively.[13] The last set corresponds to an oracle experiment in which the resolver has access to perfect SC information. Rows 3-10 of Table 7 can be interpreted in a similar manner.

From Table 6, we can see that (1) in comparison to the baseline, F-measure increases significantly in the five cases where at least one of the KSs is employed as a constraint by the resolver, and such improvements stem mainly from significant gains in precision; (2) in these five cases, the resolvers that use SCs induced by DL and SVM achieve significantly higher F-measure scores than their counterparts that rely on Soon's method for SC determination; and (3) none of the resolvers appears to benefit from *SCA* information whenever *mention* is used as a constraint.

Moreover, note that even with perfectly computed SC information, the performance of the baseline system does not improve when neither *MD* nor *SCA* is employed as a constraint. These results provide further evidence that the decision tree learner is not exploiting these two semantic KSs in an optimal manner, whether they are computed automatically or perfectly. Hence, in machine learning for coreference resolution, it is important to determine not only *what* linguistic KSs to use, but also *how* to use them.

While the coreference results in Table 6 seem to suggest that *SCA* and *mention* should be employed as constraints, the resolution results in Table 7 sug-

[13] Results using other learners are not shown due to space limitations. DL and SVM are chosen simply because they achieve the highest SC classification accuracies on the ACE training set.

gest that *SCA* is better encoded as a feature. Specifically, (1) in comparison to the baseline, the accuracy of common NP resolution improves by about 5-8% when *SCA* is encoded as a feature; and (2) whenever *SCA* is employed as a feature, the overall resolution accuracy is significantly higher for resolvers that use SCs induced by DL and SVM than those that rely on Soon's method for SC determination, with improvements in resolution observed on all three NP types.

Overall, these results provide suggestive evidence that both KSs are useful for learning-based coreference resolution. In particular, *mention* should be employed as a constraint, whereas *SCA* should be used as a feature. Interestingly, this is consistent with the results that we obtained when the resolver has access to perfect SC information (see Table 6), where the highest F-measure is achieved by employing *mention* as a constraint and *SCA* as a feature.

5 Conclusions

We have shown that (1) both *mention* and *SCA* can be usefully employed to improve the performance of a learning-based coreference system, and (2) employing SC knowledge induced in a supervised manner enables a resolver to achieve better performance than employing SC knowledge computed by Soon et al.'s simple method. In addition, we found that the MUC scoring program is unable to reveal the usefulness of the *SCA* KS, which, when encoded as a feature, substantially improves the accuracy of common NP resolution. This underscores the importance of reporting both resolution accuracy and clustering-level accuracy when analyzing the performance of a coreference resolver.

References

D. Bean and E. Riloff. 2004. Unsupervised learning of contextual role knowledge for coreference resolution. In *Proc. of HLT/NAACL*, pages 297–304.

D. M. Bikel, R. Schwartz, and R. M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning* 34(1–3):211–231.

C.-C. Chang and C.-J. Lin, 2001. LIBSVM: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *Proc. of EMNLP/VLC*.

W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. 2004. TiMBL: Tilburg Memory Based Learner, version 5.1, Reference Guide. ILK Technical Report.

H. Daumé III and D. Marcu. 2005. A large-scale exploration of effective global features for a joint entity detection and tracking model. In *Proc. of HLT/EMNLP*, pages 97–104.

R. Florian, H. Jing, N. Kambhatla, and I. Zitouni. 2006. Factorizing complex models: A case study in mention detection. In *Proc. of COLING/ACL*, pages 473–480.

M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. of COLING*.

H. Ji, D. Westbrook, and R. Grishman. 2005. Using semantic relations to refine coreference decisions. In *Proc. of HLT/EMNLP*, pages 17–24.

A. Kehler, D. Appelt, L. Taylor, and A. Simma. 2004. The (non)utility of predicate-argument frequencies for pronoun interpretation. In *Proc. of NAACL*, pages 289–296.

D. Lin. 1998a. Automatic retrieval and clustering of similar words. In *Proc. of COLING/ACL*, pages 768–774.

D. Lin. 1998b. Dependency-based evaluation of MINIPAR. In *Proc. of the LREC Workshop on the Evaluation of Parsing Systems*, pages 48–56.

D. Lin. 1998c. Using collocation statistics in information extraction. In *Proc. of MUC-7*.

X. Luo, A. Ittycheriah, H. Jing, N. Kambhatla, and S. Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the Bell tree. In *Proc. of the ACL*.

K. Markert and M. Nissim. 2005. Comparing knowledge sources for nominal anaphora resolution. *Computational Linguistics*, 31(3):367–402.

R. Mitkov. 2002. *Anaphora Resolution*. Longman.

R. Mitkov. 1998. Robust pronoun resolution with limited knowledge. In *Proc. of COLING/ACL*, pages 869–875.

V. Ng and C. Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proc. of the ACL*.

E. W. Noreen. 1989. *Computer Intensive Methods for Testing Hypothesis: An Introduction*. John Wiley & Sons.

M. Poesio, R. Mehta, A. Maroudas, and J. Hitzeman. 2004. Learning to resolve bridging references. In *Proc. of the ACL*.

S. P. Ponzetto and M. Strube. 2006. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Proc. of HLT/NAACL*, pages 192–199.

J. R. Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

W. M. Soon, H. T. Ng, and D. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

J. Tetreault. 2001. A corpus-based evaluation of centering and pronoun resolution. *Computational Linguistics*, 27(4).

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proc. of MUC-6*, pages 45–52.

R. Weischedel and A. Brunstein. 2005. BBN pronoun coreference and entity type corpus. Linguistica Data Consortium.

X. Yang, G. Zhou, J. Su, and C. L. Tan. 2003. Coreference resolution using competitive learning approach. In *Proc. of the ACL*, pages 176–183.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of the ACL*.

Generating a Table-of-Contents

S.R.K. Branavan, Pawan Deshpande and **Regina Barzilay**
Massachusetts Institute of Technology
{branavan, pawand, regina}@csail.mit.edu

Abstract

This paper presents a method for the automatic generation of a table-of-contents. This type of summary could serve as an effective navigation tool for accessing information in long texts, such as books. To generate a coherent table-of-contents, we need to capture both global dependencies across different titles in the table and local constraints within sections. Our algorithm effectively handles these complex dependencies by factoring the model into local and global components, and incrementally constructing the model's output. The results of automatic evaluation and manual assessment confirm the benefits of this design: our system is consistently ranked higher than non-hierarchical baselines.

1 Introduction

Current research in summarization focuses on processing short articles, primarily in the news domain. While in practice the existing summarization methods are not limited to this material, they are not universal: texts in many domains and genres cannot be summarized using these techniques. A particularly significant challenge is the summarization of longer texts, such as books. The requirement for high compression rates and the increased need for the preservation of contextual dependencies between summary sentences places summarization of such texts beyond the scope of current methods.

In this paper, we investigate the automatic generation of *tables-of-contents*, a type of indicative summary particularly suited for accessing information in long texts. A typical table-of-contents lists topics described in the source text and provides information about their location in the text. The hierarchical organization of information in the table further refines information access by specifying the relations between different topics and providing rich contextual information during browsing. Commonly found in books, tables-of-contents can also facilitate access to other types of texts. For instance, this type of summary could serve as an effective navigation tool for understanding a long, unstructured transcript for an academic lecture or a meeting.

Given a text, our goal is to generate a tree wherein a node represents a segment of text and a title that summarizes its content. This process involves two tasks: the hierarchical segmentation of the text, and the generation of informative titles for each segment. The first task can be addressed by using the hierarchical structure readily available in the text (e.g., chapters, sections and subsections) or by employing existing topic segmentation algorithms (Hearst, 1994). In this paper, we take the former approach. As for the second task, a naive approach would be to employ existing methods of title generation to each segment, and combine the results into a tree structure.

However, the latter approach cannot guarantee that the generated table-of-contents forms a coherent representation of the entire text. Since titles of different segments are generated in isolation, some of the generated titles may be repetitive. Even non-repetitive titles may not provide sufficient information to discriminate between the content of one seg-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 544–551,
Prague, Czech Republic, June 2007.

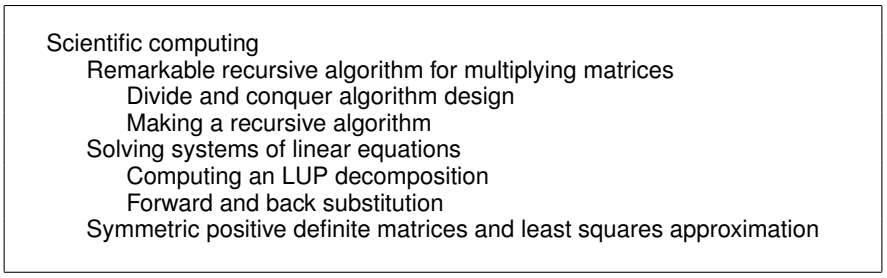


Figure 1: A fragment of a table-of-contents generated by our method.

ment and another. Therefore, it is essential to generate an entire table-of-contents tree in a concerted fashion.

This paper presents a hierarchical discriminative approach for table-of-contents generation. Figure 1 shows a fragment of a table-of-contents automatically generated by this algorithm. Our method has two important points of departure from existing techniques. First, we introduce a structured discriminative model for table-of-contents generation that accounts for a wide range of phrase-based and collocational features. The flexibility of this model results in improved summary quality. Second, our model captures both global dependencies across different titles in the tree and local dependencies within sections. We decompose the model into local and global components that handle different classes of dependencies. We further reduce the search space through incremental construction of the model's output by considering only the promising parts of the decision space.

We apply our method to process a 1,180 page algorithms textbook. To assess the contribution of our hierarchical model, we compare our method with state-of-the-art methods that generate each segment title independently.[1] The results of automatic evaluation and manual assessment of title quality show that the output of our system is consistently ranked higher than that of non-hierarchical baselines.

2 Related Work

Although most current research in summarization focuses on newspaper articles, a number of approaches have been developed for processing longer texts. Most of these approaches are tailored to a particular domain, such as medical literature or scientific articles. By making strong assumptions about the input structure and the desired format of the output, these methods achieve a high compression rate while preserving summary coherence. For instance, Teufel and Moens (2002) summarize scientific articles by selecting rhetorical elements that are commonly present in scientific abstracts. Elhadad and McKeown (2001) generate summaries of medical articles by following a certain structural template in content selection and realization.

Our work, however, is closer to domain-independent methods for summarizing long texts. Typically, these approaches employ topic segmentation to identify a list of topics described in a document, and then produce a summary for each part (Boguraev and Neff, 2000; Angheluta et al., 2002). In contrast to our method, these approaches perform either sentence or phrase extraction, rather than summary generation. Moreover, extraction for each segment is performed in isolation, and global constraints on the summary are not enforced.

Finally, our work is also related to research on title generation (Banko et al., 2000; Jin and Hauptmann, 2001; Dorr et al., 2003). Since work in this area focuses on generating titles for one article at a time (e.g., newspaper reports), the issue of hierarchical generation, which is unique to our task, does not arise. However, this is not the only novel aspect of the proposed approach. Our model learns title generation in a fully discriminative framework, in contrast to the commonly used noisy-channel model. Thus, instead of independently modeling the selection and grammaticality constraints, we learn both types of features in a single framework. This joint training regime supports greater flexibility in modeling feature interaction.

[1] The code and feature vector data for our model and the baselines are available at http://people.csail.mit.edu/branavan/code/toc.

3 Problem Formulation

We formalize the problem of table-of-contents generation as a supervised learning task where the goal is to map a tree of text segments S to a tree of titles T. A segment may correspond to a chapter, section or subsection.

Since the focus of our work is on the generation aspect of table-of-contents construction, we assume that the hierarchical segmentation of a text is provided in the input. This division can either be automatically computed using one of the many available text segmentation algorithms (Hearst, 1994), or it can be based on demarcations already present in the input (e.g., paragraph markers).

During training, the algorithm is provided with a set of pairs (S^i, T^i) for $i = 1, \ldots, p$, where S^i is the i^{th} tree of text segments, and T^i is the table-of-contents for that tree. During testing, the algorithm generates tables-of-contents for unseen trees of text segments.

We also assume that during testing the desired title length is provided as a parameter to the algorithm.

4 Algorithm

To generate a coherent table-of-contents, we need to take into account multiple constraints: the titles should be grammatical, they should adequately represent the content of their segments, and the table-of-contents as a whole should clearly convey the relations between the segments. Taking a discriminative approach for modeling this task would allow us to achieve this goal: we can easily integrate a range of constraints in a flexible manner. Since the number of possible labels (i.e., tables-of-contents) is prohibitively large and the labels themselves exhibit a rich internal structure, we employ a structured discriminative model that can easily handle complex dependencies. Our solution relies on two orthogonal strategies to balance the tractability and the richness of the model. First, we factor the model into local and global components. Second, we incrementally construct the output of each component using a search-based discriminative algorithm. Both of these strategies have the effect of intelligently pruning the decision space.

Our model factorization is driven by the different types of dependencies which are captured by the two components. The first model is *local*: for each segment, it generates a list of candidate titles ranked by their individual likelihoods. This model focuses on grammaticality and word selection constraints, but it does not consider relations among different titles in the table-of-contents. These latter dependencies are captured in the *global* model that constructs a table-of-contents by selecting titles for each segment from the available candidates. Even after this factorization, the decision space for each model is large: for the local model, it is exponential in the length of the segment title, and for the global model it is exponential in the size of the tree.

Therefore, we construct the output for each of these models *incrementally* using beam search. The algorithm maintains the most promising partial output structures, which are extended at every iteration. The model incorporates this decoding procedure into the training process, thereby learning model parameters best suited for the specific decoding algorithm. Similar models have been successfully applied in the past to other tasks including parsing (Collins and Roark, 2004), chunking (Daumé and Marcu, 2005), and machine translation (Cowan et al., 2006).

4.1 Model Structure

The model takes as input a tree of text segments S. Each segment $s \in S$ and its title z are represented as a *local feature vector* $\Phi_{\mathbf{loc}}(s, z)$. Each component of this vector stores a numerical value. This feature vector can track any feature of the segment s together with its title z. For instance, the i^{th} component of this vector may indicate whether the bigram $(z[j]z[j+1])$ occurs in s, where $z[j]$ is the j^{th} word in z:

$$(\Phi_{\mathbf{loc}}(s, z))_i = \begin{cases} 1 & \text{if } (z[j]z[j+1]) \in s \\ 0 & \text{otherwise} \end{cases}$$

In addition, our model captures dependencies among *multiple titles* that appear in the same table-of-contents. We represent a tree of segments S paired with titles T with the *global feature vector* $\Phi_{\mathbf{glob}}(S, T)$. The components here are also numerical features. For example, the i^{th} component of the vector may indicate whether a title is repeated in the table-of-contents T:

$$(\Phi_{\mathbf{glob}}(S,T))_i = \begin{cases} 1 & \text{repeated title} \\ 0 & \text{otherwise} \end{cases}$$

Our model constructs a table-of-contents in two basic steps:

Step One The goal of this step is to generate a list of k candidate titles for each segment $s \in S$. To do so, for each possible title z, the model maps the feature vector $\Phi_{\mathbf{loc}}(s,z)$ to a real number. This mapping can take the form of a linear model,

$$\Phi_{\mathbf{loc}}(s,z) \cdot \alpha_{\text{loc}}$$

where α_{loc} is the local parameter vector.

Since the number of possible titles is exponential, we cannot consider all of them. Instead, we prune the decision space by incrementally constructing promising titles. At each iteration j, the algorithm maintains a beam Q of the top k partially generated titles of length j. During iteration $j+1$, a new set of candidates is grown by appending a word from s to the right of each member of the beam Q. We then sort the entries in Q: $z_1, z_2, \ldots$ such that $\Phi_{\mathbf{loc}}(s,z_i) \cdot \alpha_{\text{loc}} \geq \Phi_{\mathbf{loc}}(s,z_{i+1}) \cdot \alpha_{\text{loc}}, \forall i$. Only the top k candidates are retained, forming the beam for the next iteration. This process continues until a title of the desired length is generated. Finally, the list of k candidates is returned.

Step Two Given a set of candidate titles $z_1, z_2, \ldots, z_k$ for each segment $s \in S$, our goal is to construct a table-of-contents T by selecting the most appropriate title from each segment's candidate list. To do so, our model computes a score for the pair (S,T) based on the global feature vector $\Phi_{\mathbf{glob}}(S,T)$:

$$\Phi_{\mathbf{glob}}(S,T) \cdot \alpha_{\text{glob}}$$

where α_{glob} is the global parameter vector.

As with the local model (step one), the number of possible tables-of-contents is too large to be considered exhaustively. Therefore, we incrementally construct a table-of-contents by traversing the tree of segments in a pre-order walk (i.e., the order in which segments appear in the text). In this case, the beam contains partially generated tables-of-contents, which are expanded by one segment title at a time. To further reduce the search space, during decoding only the top five candidate titles for a segment are given to the global model.

4.2 Training the Model

Training for Step One We now describe how the local parameter vector α_{loc} is estimated from training data. We are given a set of training examples (s^i, y^i) for $i = 1, \ldots, l$, where s^i is the i^{th} text segment, and y^i is the title of this segment.

This linear model is learned using a variant of the incremental perceptron algorithm (Collins and Roark, 2004; Daumé and Marcu, 2005). This online algorithm traverses the training set multiple times, updating the parameter vector α_{loc} after each training example in case of mis-predictions. The algorithm encourages a setting of the parameter vector α_{loc} that assigns the highest score to the feature vector associated with the correct title.

The pseudo-code of the algorithm is shown in Figure 2. Given a text segment s and the corresponding title y, the training algorithm maintains a beam Q containing the top k partial titles of length j. The beam is updated on each iteration using the functions GROW and PRUNE. For every word in segment s and for every partial title in Q, GROW creates a new title by appending this word to the title. PRUNE retains only the top ranked candidates based on the scoring function $\Phi_{\mathbf{loc}}(s,z) \cdot \alpha_{\text{loc}}$. If $y[1 \ldots j]$ (i.e., the prefix of y of length j) is not in the modified beam Q, then α_{loc} is updated[2] as shown in line 4 of the pseudo-code in Figure 2. In addition, Q is replaced with a beam containing only $y[1 \ldots j]$ (line 5). This process is performed $|y|$ times. We repeat this process for all training examples over 50 training iterations. [3]

Training for Step Two To train the global parameter vector α_{glob}, we are given training examples (S^i, T^i) for $i = 1, \ldots, p$, where S^i is the i^{th} tree of text segments, and T^i is the table-of-contents for that tree. However, we cannot directly use these tables-of-contents for training our global model: since this model selects one of the candidate titles $z_1^i, \ldots, z_k^i$ returned by the local model, the true title of the segment may not be among these candidates. Therefore, to determine a new target title for the segment, we need to identify the title in the set of candidates

[2] If the word in the j^{th} position of y does not occur in s, then the parameter update is not performed.

[3] For decoding, α_{loc} is averaged over the training iterations as in Collins and Roark (2004).

s – segment text.
y – segment title.
$y[1 \dots j]$ – prefix of y of length j.
Q – beam containing partial titles.

1. for $j = 1 \dots |y|$
2. $\quad Q = \text{PRUNE}(\text{GROW}(s, Q))$
3. $\quad$ if $y[1 \dots j] \notin Q$
4. $\quad\quad \alpha_{\text{loc}} = \alpha_{\text{loc}} + \Phi_{\mathbf{loc}}(s, y[1 \dots j]) - \sum_{z \in Q} \frac{\Phi_{\mathbf{loc}}(s,z)}{|Q|}$
5. $\quad\quad Q = \{y[1 \dots j]\}$

Figure 2: The training algorithm for the local model.

that is closest to the true title.

We employ the L_1 distance measure to compare the content word overlap between two titles.[4] For each input (S, T), and each segment $s \in S$, we identify the segment title closest in the L_1 measure to the true title y[5]:

$$z^* = \arg\min_i L_1(z_i, y)$$

Once all the training targets in the corpus have been identified through this procedure, the global linear model $\Phi_{\mathbf{glob}}(S, T) \cdot \alpha_{\text{glob}}$ is learned using the same perceptron algorithm as in step one. Rather than maintaining the beam of partially generated titles, the beam Q holds partially generated tables-of-contents. Also, the loop in line 1 of Figure 2 iterates over segment titles rather than words. The global model is trained over 200 iterations.

5 Features

Local Features Our local model aims to generate titles which adequately represent the meaning of the segment and are grammatical. Selection and contextual preferences are encoded in the local features. The features that capture selection constraints are specified at the word level, and contextual features are expressed at the word sequence level.

The selection features capture the position of the word, its TF*IDF, and part-of-speech information. In addition, they also record whether the word occurs in the body of neighboring segments. We also generate conjunctive features by combining features of different types.

The contextual features record the bigram and trigram language model scores, both for words and for part-of-speech tags. The trigram scores are averaged over the title. The language models are trained using the SRILM toolkit. Another type of contextual feature models the collocational properties of noun phrases in the title. This feature aims to eliminate generic phrases, such as *"the following section"* from the generated titles.[6] To achieve this effect, for each noun phrase in the title, we measure the ratio of their frequency in the segment to their frequency in the corpus.

Segment has the same title as its sibling
Segment has the same title as its parent
Two adjacent sibling titles have the same head
Two adjacent sibling titles start with the same word
Rank given to the title by the local model

Table 1: Examples of global features.

Global Features Our global model describes the interaction between different titles in the tree (See Table 1). These interactions are encoded in three types of global features. The first type of global feature indicates whether titles in the tree are redundant at various levels of the tree structure. The second type of feature encourages parallel constructions within the same tree. For instance, titles of adjoining segments may be verbalized as noun phrases with the same head (e.g., *"Bubble sort algorithm"*, *"Merge sort algorithm"*). We capture this property by comparing words that appear in certain positions in adjacent sibling titles. Finally, our global model also uses the rank of the title provided by the local model. This feature enables the global model to account for the preferences of the local model in the title selection process.

6 Evaluation Set-Up

Data We apply our method to an undergraduate algorithms textbook. For detailed statistics on the data see Table 2. We split its table-of-contents into a set

[4] This measure is close to ROUGE-1 which in addition considers the overlap in auxiliary words.

[5] In the case of ties, one of the titles is picked arbitrarily.

[6] Unfortunately, we could not use more sophisticated syntactic features due to the low accuracy of statistical parsers on our corpus.

Number of Titles	540
Number of Trees	39
Tree Depth	4
Number of Words	269,650
Avg. Title Length	3.64
Avg. Branching	3.29
Avg. Title Duplicates	21

Table 2: Statistics on the corpus used in the experiments.

of independent subtrees. Given a table-of-contents of depth n with a root branching factor of r, we generate r subtrees, with a depth of at most $n-1$. We randomly select 80% of these trees for training, and the rest are used for testing. In our experiments, we use ten different randomizations to compensate for the small number of available trees.

Admittedly, this method of generating training and testing data omits some dependencies at the level of the table-of-contents as a whole. However, the subtrees used in our experiments still exhibit a sufficiently deep hierarchical structure, rich with contextual dependencies.

Baselines As an alternative to our hierarchical discriminative method, we consider three baselines that build a table-of-contents by generating a title for each segment individually, without taking into account the tree structure, and one hierarchical generative baseline. The first method generates a title for a segment by selecting the noun phrase from that segment with the highest TF*IDF. This simple method is commonly used to generate keywords for browsing applications in information retrieval, and has been shown to be effective for summarizing technical content (Wacholder et al., 2001).

The second baseline is based on the noisy-channel generative (flat generative, FG) model proposed by Banko et al., (2000). Similar to our local model, this method captures both selection and grammatical constraints. However, these constraints are modeled separately, and then combined in a generative framework.

We use our local model (Flat Discriminative model, FD) as the third baseline. Like the second baseline, this model omits global dependencies, and only focuses on features that capture relations within individual segments.

In the hierarchical generative (HG) baseline we run our global model on the ranked list of titles produced for each section by the noisy-channel generative model.

The last three baselines and our algorithm are provided with the title length as a parameter. In our experiments, the algorithms use the reference title length.

Experimental Design: Comparison with reference tables-of-contents Reference based evaluation is commonly used to assess the quality of machine-generated headlines (Wang et al., 2005). We compare our system's output with the table-of-contents from the textbook using ROUGE metrics. We employ a publicly available software package,[7] with all the parameters set to default values.

Experimental Design: Human assessment The judges were each given 30 segments randomly selected from a set of 359 test segments. For each test segment, the judges were presented with its text, and 3 alternative titles consisting of the reference and the titles produced by the hierarchical discriminative model, and the best performing baseline. In addition, the judges had access to all of the segments in the book. A total of 498 titles for 166 unique segments were ranked. The system identities were hidden from the judges, and the titles were presented in random order. The judges ranked the titles based on how well they represent the content of the segment. Titles were ranked equal if they were judged to be equally representative of the segment.

Six people participated in this experiment. All the participants were graduate students in computer science who had taken the algorithms class in the past and were reasonably familiar with the material.

7 Results

Figure 3 shows fragments of the tables-of-contents generated by our method and the four baselines along with the reference counterpart. These extracts illustrate three general phenomena that we observed in the test corpus. First, the titles produced by keyword extraction exhibit a high degree of redundancy. In fact, 40% of the titles produced by this method are repeated more than once in the table-of-contents. In

[7] http://www.isi.edu/licensed-sw/see/rouge/

Reference:	**Flat Generative:**	**Flat Discriminative:**
hash tables direct address tables hash tables collision resolution by chaining analysis of hashing with chaining open addressing linear probing quadratic probing double hashing	linked list worst case time wasted space worst case running time to show that there are dynamic set occupied slot quadratic function double hashing	dictionary operations universe of keys computer memory element in the list hash table with load factor hash table hash function hash function double hashing
Keyword Extraction:	**Hierarchical Generative:**	**Hierarchical Discriminative:**
hash table dynamic set hash function worst case expected number hash table hash function hash table double hashing	dictionary operations worst case time wasted space worst case running time to show that there are collision resolution linear time quadratic function double hashing	dictionary operations direct address table computer memory worst case running time hash table with load factor address table hash function quadratic probing double hashing

Figure 3: Fragments of tables-of-contents generated by our method and the four baselines along with the corresponding reference.

	Rouge-1	Rouge-L	Rouge-W	Full Match
HD	**0.256**	**0.249**	**0.216**	**13.5**
FD	0.241	0.234	0.203	13.1
HG	0.139	0.133	0.117	5.8
FG	0.094	0.090	0.079	4.1
Keyword	0.168	0.168	0.157	6.3

Table 3: Title quality as compared to the reference for the hierarchical discriminative (HD), flat discriminative (FD), hierarchical generative (HG), flat generative (FG) and Keyword models. The improvement given by HD over FD in all three Rouge measures is significant at $p \leq 0.03$ based on the Sign test.

	better	worse	equal
HD vs. FD	68	32	49
Reference vs. HD	115	13	22
Reference vs. FD	123	7	20

Table 4: Overall pairwise comparisons of the rankings given by the judges. The improvement in title quality given by HD over FD is significant at $p \leq 0.0002$ based on the Sign test.

contrast, our method yields 5.5% of the titles as duplicates, as compared to 9% in the reference table-of-contents.[8]

Second, the fragments show that the two discriminative models — Flat and Hierarchical — have a number of common titles. However, adding global dependencies to rerank titles generated by the local model changes 30% of the titles in the test set.

Comparison with reference tables-of-contents Table 3 shows the average ROUGE scores over the ten randomizations for the five automatic methods. The hierarchical discriminative method consistently outperforms the four baselines according to all ROUGE metrics.

At the same time, these results also show that only a small ratio of the automatically generated titles are identical to the reference ones. In some cases, the machine-generated titles are very close in meaning to the reference, but are verbalized differently. Examples include pairs such as *("Minimum Spanning Trees", "Spanning Tree Problem")* and *("Wallace Tree", "Multiplication Circuit").*[9] While measures like ROUGE can capture the similarity in the first pair, they cannot identify semantic proximity

[8]Titles such as *"Analysis"* and *"Chapter Outline"* are repeated multiple times in the text.

[9]A Wallace Tree is a circuit that multiplies two integers.

between the titles in the second pair. Therefore, we supplement the results of this experiment with a manual assessment of title quality as described below.

Human assessment We analyze the human ratings by considering pairwise comparisons between the models. Given two models, A and B, three outcomes are possible: A is better than B, B is better than A, or they are of equal quality. The results of the comparison are summarized in Table 4. These results indicate that using hierarchical information yields statistically significant improvement (at $p \leq 0.0002$ based on the Sign test) over a flat counterpart.

8 Conclusion and Future Work

This paper presents a method for the automatic generation of a table-of-contents. The key strength of our method lies in its ability to track dependencies between generation decisions across different levels of the tree structure. The results of automatic evaluation and manual assessment confirm the benefits of joint tree learning: our system is consistently ranked higher than non-hierarchical baselines.

We also plan to expand our method for the task of slide generation. Like tables-of-contents, slide bullets are organized in a hierarchical fashion and are written in relatively short phrases. From the language viewpoint, however, slides exhibit more variability and complexity than a typical table-of-contents. To address this challenge, we will explore more powerful generation methods that take into account syntactic information.

Acknowledgments

The authors acknowledge the support of the National Science Foundation (CAREER grant IIS-0448168 and grant IIS-0415865). We would also like to acknowledge the many people who took part in human evaluations. Thanks to Michael Collins, Benjamin Snyder, Igor Malioutov, Jacob Eisenstein, Luke Zettlemoyer, Terry Koo, Erdong Chen, Zoran Dzunic and the anonymous reviewers for helpful comments and suggestions. Any opinions, findings, conclusions or recommendations expressed above are those of the authors and do not necessarily reflect the views of the NSF.

References

Roxana Angheluta, Rik De Busser, and Marie-Francine Moens. 2002. The use of topic segmentation for automatic summarization. In *Proceedings of the ACL-2002 Workshop on Automatic Summarization*.

Michele Banko, Vibhu O. Mittal, and Michael J. Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of the ACL*, pages 318–325.

Branimir Boguraev and Mary S. Neff. 2000. Discourse segmentation in aid of document summarization. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, pages 3004–3014.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the ACL*, pages 111–118.

Brooke Cowan, Ivona Kucerova, and Michael Collins. 2006. A discriminative model for tree-to-tree translation. In *Proceedings of the EMNLP*, pages 232–241.

Hal Daumé and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the ICML*, pages 169–176.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: a parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 on Text summarization workshop*, pages 1–8.

Noemie Elhadad and Kathleen R. McKeown. 2001. Towards generating patient specific summaries of medical articles. In *Proceedings of NAACL Workshop on Automatic Summarization*, pages 31–39.

Marti Hearst. 1994. Multi-paragraph segmentation of expository text. In *Proceedings of the ACL*, pages 9–16.

Rong Jin and Alexander G. Hauptmann. 2001. Automatic title generation for spoken broadcast news. In *Proceedings of the HLT*, pages 1–3.

Simone Teufel and Marc Moens. 2002. Summarizing scientific articles: Experiments with relevance and rhetorical status. *Computational Linguistics*, 28(4):409–445.

Nina Wacholder, David K. Evans, and Judith Klavans. 2001. Automatic identification and organization of index terms for interactive browsing. In *JCDL*, pages 126–134.

R. Wang, J. Dunnion, and J. Carthy. 2005. Machine learning approach to augmenting news headline generation. In *Proceedings of the IJCNLP*.

Towards an Iterative Reinforcement Approach for Simultaneous Document Summarization and Keyword Extraction

Xiaojun Wan **Jianwu Yang** **Jianguo Xiao**
Institute of Computer Science and Technology
Peking University, Beijing 100871, China
{wanxiaojun,yangjianwu,xiaojianguo}@icst.pku.edu.cn

Abstract

Though both document summarization and keyword extraction aim to extract concise representations from documents, these two tasks have usually been investigated independently. This paper proposes a novel iterative reinforcement approach to simultaneously extracting summary and keywords from single document under the assumption that the summary and keywords of a document can be mutually boosted. The approach can naturally make full use of the reinforcement between sentences and keywords by fusing three kinds of relationships between sentences and words, either homogeneous or heterogeneous. Experimental results show the effectiveness of the proposed approach for both tasks. The corpus-based approach is validated to work almost as well as the knowledge-based approach for computing word semantics.

1 Introduction

Text summarization is the process of creating a compressed version of a given document that delivers the main topic of the document. Keyword extraction is the process of extracting a few salient words (or phrases) from a given text and using the words to represent the text. The two tasks are similar in essence because they both aim to extract concise representations for documents. Automatic text summarization and keyword extraction have drawn much attention for a long time because they both are very important for many text applications, including document retrieval, document clustering, etc. For example, keywords of a document can be used for document indexing and thus benefit to improve the performance of document retrieval, and document summary can help to facilitate users to browse the search results and improve users' search experience.

Text summaries and keywords can be either query-relevant or generic. Generic summary and keyword should reflect the main topics of the document without any additional clues and prior knowledge. In this paper, we focus on generic document summarization and keyword extraction for single documents.

Document summarization and keyword extraction have been widely explored in the natural language processing and information retrieval communities. A series of workshops and conferences on automatic text summarization (e.g. SUMMAC, DUC and NTCIR) have advanced the technology and produced a couple of experimental online systems. In recent years, graph-based ranking algorithms have been successfully used for document summarization (Mihalcea and Tarau, 2004, 2005; ErKan and Radev, 2004) and keyword extraction (Mihalcea and Tarau, 2004). Such algorithms make use of "voting" or "recommendations" between sentences (or words) to extract sentences (or keywords). Though the two tasks essentially share much in common, most algorithms have been developed particularly for either document summarization or keyword extraction.

Zha (2002) proposes a method for simultaneous keyphrase extraction and text summarization by using only the heterogeneous sentence-to-word relationships. Inspired by this, we aim to take into account all the three kinds of relationships among sentences and words (i.e. the homogeneous relationships between words, the homogeneous relationships between sentences, and the heterogeneous relationships between words and sentences) in

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 552–559,
Prague, Czech Republic, June 2007.

a unified framework for both document summarization and keyword extraction. The importance of a sentence (word) is determined by both the importance of related sentences (words) and the importance of related words (sentences). The proposed approach can be considered as a generalized form of previous graph-based ranking algorithms and Zha's work (Zha, 2002).

In this study, we propose an iterative reinforcement approach to realize the above idea. The proposed approach is evaluated on the DUC2002 dataset and the results demonstrate its effectiveness for both document summarization and keyword extraction. Both knowledge-based approach and corpus-based approach have been investigated to compute word semantics and they both perform very well.

The rest of this paper is organized as follows: Section 2 introduces related works. The details of the proposed approach are described in Section 3. Section 4 presents and discusses the evaluation results. Lastly we conclude our paper in Section 5.

2 Related Works

2.1 Document Summarization

Generally speaking, single document summarization methods can be either extraction-based or abstraction-based and we focus on extraction-based methods in this study.

Extraction-based methods usually assign a saliency score to each sentence and then rank the sentences in the document. The scores are usually computed based on a combination of statistical and linguistic features, including term frequency, sentence position, cue words, stigma words, topic signature (Hovy and Lin, 1997; Lin and Hovy, 2000), etc. Machine learning methods have also been employed to extract sentences, including unsupervised methods (Nomoto and Matsumoto, 2001) and supervised methods (Kupiec et al., 1995; Conroy and O'Leary, 2001; Amini and Gallinari, 2002; Shen et al., 2007). Other methods include maximal marginal relevance (MMR) (Carbonell and Goldstein, 1998), latent semantic analysis (LSA) (Gong and Liu, 2001). In Zha (2002), the mutual reinforcement principle is employed to iteratively extract key phrases and sentences from a document.

Most recently, graph-based ranking methods, including TextRank ((Mihalcea and Tarau, 2004, 2005) and LexPageRank (ErKan and Radev, 2004) have been proposed for document summarization. Similar to Kleinberg's HITS algorithm (Kleinberg, 1999) or Google's PageRank (Brin and Page, 1998), these methods first build a graph based on the similarity between sentences in a document and then the importance of a sentence is determined by taking into account global information on the graph recursively, rather than relying only on local sentence-specific information.

2.2 Keyword Extraction

Keyword (or keyphrase) extraction usually involves assigning a saliency score to each candidate keyword by considering various features. Krulwich and Burkey (1996) use heuristics to extract keyphrases from a document. The heuristics are based on syntactic clues, such as the use of italics, the presence of phrases in section headers, and the use of acronyms. Muñoz (1996) uses an unsupervised learning algorithm to discover two-word keyphrases. The algorithm is based on Adaptive Resonance Theory (ART) neural networks. Steier and Belew (1993) use the mutual information statistics to discover two-word keyphrases.

Supervised machine learning algorithms have been proposed to classify a candidate phrase into either keyphrase or not. GenEx (Turney, 2000) and Kea (Frank et al., 1999; Witten et al., 1999) are two typical systems, and the most important features for classifying a candidate phrase are the frequency and location of the phrase in the document. More linguistic knowledge (such as syntactic features) has been explored by Hulth (2003). More recently, Mihalcea and Tarau (2004) propose the TextRank model to rank keywords based on the co-occurrence links between words.

3 Iterative Reinforcement Approach

3.1 Overview

The proposed approach is intuitively based on the following assumptions:

Assumption 1: A sentence should be salient if it is heavily linked with other salient sentences, and a word should be salient if it is heavily linked with other salient words.

Assumption 2: A sentence should be salient if it contains many salient words, and a word should be salient if it appears in many salient sentences.

The first assumption is similar to PageRank which makes use of mutual "recommendations"

between homogeneous objects to rank objects. The second assumption is similar to HITS if words and sentences are considered as authorities and hubs respectively. In other words, the proposed approach aims to fuse the ideas of PageRank and HITS in a unified framework.

In more detail, given the heterogeneous data points of sentences and words, the following three kinds of relationships are fused in the proposed approach:

SS-Relationship: It reflects the homogeneous relationships between sentences, usually computed by their content similarity.

WW-Relationship: It reflects the homogeneous relationships between words, usually computed by knowledge-based approach or corpus-based approach.

SW-Relationship: It reflects the heterogeneous relationships between sentences and words, usually computed as the relative importance of a word in a sentence.

Figure 1 gives an illustration of the relationships.

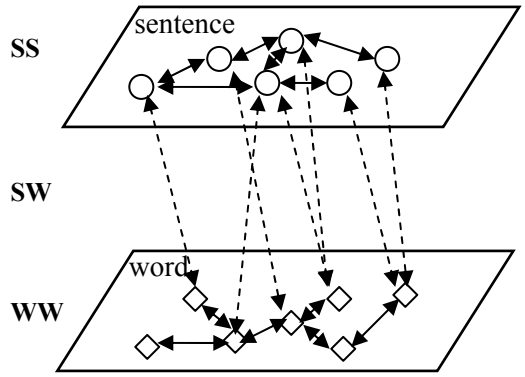


Figure 1. Illustration of the Relationships

The proposed approach first builds three graphs to reflect the above relationships respectively, and then iteratively computes the saliency scores of the sentences and words based on the graphs. Finally, the algorithm converges and each sentence or word gets its saliency score. The sentences with high saliency scores are chosen into the summary, and the words with high saliency scores are combined to produce the keywords.

3.2 Graph Building

3.2.1 Sentence-to-Sentence Graph (SS-Graph)

Given the sentence collection $S=\{s_i \mid 1\le i\le m\}$ of a document, if each sentence is considered as a node, the sentence collection can be modeled as an undirected graph by generating an edge between two sentences if their content similarity exceeds 0, i.e. an undirected link between s_i and s_j ($i\neq j$) is constructed and the associated weight is their content similarity. Thus, we construct an undirected graph G_{SS} to reflect the homogeneous relationship between sentences. The content similarity between two sentences is computed with the cosine measure. We use an adjacency matrix $\boldsymbol{U}$ to describe G_{SS} with each entry corresponding to the weight of a link in the graph. $\boldsymbol{U}=[U_{ij}]_{m\times m}$ is defined as follows:

$$U_{ij}=\begin{cases}\dfrac{\vec{s}_i \times \vec{s}_j}{\|\vec{s}_i\| \times \|\vec{s}_j\|}, & if\ i \neq j \\ 0, & otherwise\end{cases} \quad (1)$$

where s_i and $\vec{s}_j$ are the corresponding term vectors of sentences s_i and s_j respectively. The weight associated with term t is calculated with $tf_t.isf_t$, where tf_t is the frequency of term t in the sentence and isf_t is the inverse sentence frequency of term t, i.e. $1+\log(N/n_t)$, where N is the total number of sentences and n_t is the number of sentences containing term t in a background corpus. Note that other measures (e.g. Jaccard, Dice, Overlap, etc.) can also be explored to compute the content similarity between sentences, and we simply choose the cosine measure in this study.

Then $\boldsymbol{U}$ is normalized to $\widetilde{\boldsymbol{U}}$ as follows to make the sum of each row equal to 1:

$$\widetilde{U}_{ij}=\begin{cases}U_{ij} \Big/ \sum_{j=1}^{m} U_{ij}, & if \sum_{j=1}^{m} U_{ij} \neq 0 \\ 0, & otherwise\end{cases} \quad (2)$$

3.2.2 Word-to-Word Graph (WW-Graph)

Given the word collection $T=\{t_j \mid 1\le j\le n\}$ of a document[1], the semantic similarity between any two words t_i and t_j can be computed using approaches that are either knowledge-based or corpus-based (Mihalcea et al., 2006).

Knowledge-based measures of word semantic similarity try to quantify the degree to which two words are semantically related using information drawn from semantic networks. WordNet (Fellbaum, 1998) is a lexical database where each

[1] The stopwords defined in the Smart system have been removed from the collection.

unique meaning of a word is represented by a synonym set or *synset*. Each synset has a gloss that defines the concept that it represents. Synsets are connected to each other through explicit semantic relations that are defined in WordNet. Many approaches have been proposed to measure semantic relatedness based on WordNet. The measures vary from simple edge-counting to attempt to factor in peculiarities of the network structure by considering link direction, relative path, and density, such as *vector*, *lesk*, *hso, lch*, *wup*, *path*, *res, lin* and *jcn* (Pedersen et al., 2004). For example, "cat" and "dog" has higher semantic similarity than "cat" and "computer". In this study, we implement the *vector* measure to efficiently evaluate the similarities of a large number of word pairs. The *vector* measure (Patwardhan, 2003) creates a co–occurrence matrix from a corpus made up of the WordNet glosses. Each content word used in a WordNet gloss has an associated context vector. Each gloss is represented by a gloss vector that is the average of all the context vectors of the words found in the gloss. Relatedness between concepts is measured by finding the cosine between a pair of gloss vectors.

Corpus-based measures of word semantic similarity try to identify the degree of similarity between words using information exclusively derived from large corpora. Such measures as mutual information (Turney 2001), latent semantic analysis (Landauer et al., 1998), log-likelihood ratio (Dunning, 1993) have been proposed to evaluate word semantic similarity based on the co-occurrence information on a large corpus. In this study, we simply choose the mutual information to compute the semantic similarity between word t_i and t_j as follows:

$$sim(t_i,t_j) = \log \frac{N \times p(t_i,t_j)}{p(t_i) \times p(t_j)} \qquad (3)$$

which indicates the degree of statistical dependence between t_i and t_j. Here, N is the total number of words in the corpus and $p(t_i)$ and $p(t_j)$ are respectively the probabilities of the occurrences of t_i and t_j, i.e. $count(t_i)/N$ and $count(t_j)/N$, where $count(t_i)$ and $count(t_j)$ are the frequencies of t_i and t_j. $p(t_i, t_j)$ is the probability of the co-occurrence of t_i and t_j within a window with a predefined size k, i.e. $count(t_i, t_j)/N$, where $count(t_i, t_j)$ is the number of the times t_i and t_j co-occur within the window.

Similar to the SS-Graph, we can build an undirected graph G_{WW} to reflect the homogeneous relationship between words, in which each node corresponds to a word and the weight associated with the edge between any different word t_i and t_j is computed by either the WordNet-based *vector* measure or the corpus-based mutual information measure. We use an adjacency matrix $\boldsymbol{V}$ to describe G_{WW} with each entry corresponding to the weight of a link in the graph. $\boldsymbol{V}= [V_{ij}]_{n\times n}$, where $V_{ij} = sim(t_i, t_j)$ if $i \neq j$ and $V_{ij}=0$ if $i=j$.

Then $\boldsymbol{V}$ is similarly normalized to $\widetilde{\boldsymbol{V}}$ to make the sum of each row equal to 1.

3.2.3 Sentence-to-Word Graph (SW-Graph)

Given the sentence collection $S=\{s_i \mid 1\le i\le m\}$ and the word collection $T=\{t_j \mid 1\le j\le n\}$ of a document, we can build a weighted bipartite graph G_{SW} from S and T in the following way: if word t_j appears in sentence s_i, we then create an edge between s_i and t_j. A nonnegative weight $aff(s_i,t_j)$ is specified on the edge, which is proportional to the importance of word t_j in sentence s_i, computed as follows:

$$aff(s_i,t_j) = \frac{tf_{t_j} \times isf_{t_j}}{\sum_{t \in s_i} tf_t \times isf_t} \qquad (4)$$

where t represents a unique term in s_i and tf_t, isf_t are respectively the term frequency in the sentence and the inverse sentence frequency.

We use an adjacency (affinity) matrix $\boldsymbol{W}=[W_{ij}]_{m\times n}$ to describe G_{SW} with each entry W_{ij} corresponding to $aff(s_i,t_j)$. Similarly, $\boldsymbol{W}$ is normalized to $\widetilde{\boldsymbol{W}}$ to make the sum of each row equal to 1. In addition, we normalize the transpose of $\boldsymbol{W}$, i.e. $\boldsymbol{W}^{\mathrm{T}}$, to $\hat{\boldsymbol{W}}$ to make the sum of each row in $\boldsymbol{W}^{\mathrm{T}}$ equal to 1.

3.3 Reinforcement Algorithm

We use two column vectors $\boldsymbol{u}=[u(s_i)]_{m\times 1}$ and $\boldsymbol{v}=[v(t_j)]_{n\times 1}$ to denote the saliency scores of the sentences and words in the specified document. The assumptions introduced in Section 3.1 can be rendered as follows:

$$u(s_i) \propto \sum_j \widetilde{U}_{ji} u(s_j) \qquad (5)$$

$$v(t_j) \propto \sum_i \widetilde{V}_{ij} v(t_i) \qquad (6)$$

$$u(s_i) \propto \sum_j \hat{W}_{ji} v(t_j) \qquad (7)$$

$$v(t_j) \propto \sum_i \widetilde{W}_{ij} u(s_i) \quad (8)$$

After fusing the above equations, we can obtain the following iterative forms:

$$u(s_i) = \alpha \sum_{j=1}^{m} \widetilde{U}_{ji} u(s_j) + \beta \sum_{j=1}^{n} \hat{W}_{ji} v(t_j) \quad (9)$$

$$v(t_j) = \alpha \sum_{i=1}^{n} \widetilde{V}_{ij} v(t_i) + \beta \sum_{i=1}^{m} \widetilde{W}_{ij} u(s_i) \quad (10)$$

And the matrix form is:

$$\boldsymbol{u} = \alpha \widetilde{\boldsymbol{U}}^T \boldsymbol{u} + \beta \hat{\boldsymbol{W}}^T \boldsymbol{v} \quad (11)$$

$$\boldsymbol{v} = \alpha \widetilde{\boldsymbol{V}}^T \boldsymbol{v} + \beta \widetilde{\boldsymbol{W}}^T \boldsymbol{u} \quad (12)$$

where α and β specify the relative contributions to the final saliency scores from the homogeneous nodes and the heterogeneous nodes and we have $\alpha+\beta=1$. In order to guarantee the convergence of the iterative form, $\boldsymbol{u}$ and $\boldsymbol{v}$ are normalized after each iteration.

For numerical computation of the saliency scores, the initial scores of all sentences and words are set to 1 and the following two steps are alternated until convergence,

1. Compute and normalize the scores of sentences:
$$\boldsymbol{u}^{(n)} = \alpha \widetilde{\boldsymbol{U}}^T \boldsymbol{u}^{(n-1)} + \beta \hat{\boldsymbol{W}}^T \boldsymbol{v}^{(n-1)},$$
$$\boldsymbol{u}^{(n)} = \boldsymbol{u}^{(n)} / \left\| \boldsymbol{u}^{(n)} \right\|_1$$
2. Compute and normalize the scores of words:
$$\boldsymbol{v}^{(n)} = \alpha \widetilde{\boldsymbol{V}}^T \boldsymbol{v}^{(n-1)} + \beta \widetilde{\boldsymbol{W}}^T \boldsymbol{u}^{(n-1)},$$
$$\boldsymbol{v}^{(n)} = \boldsymbol{v}^{(n)} / \left\| \boldsymbol{v}^{(n)} \right\|_1$$

where $\boldsymbol{u}^{(n)}$ and $\boldsymbol{v}^{(n)}$ denote the vectors computed at the n-th iteration.

Usually the convergence of the iteration algorithm is achieved when the difference between the scores computed at two successive iterations for any sentences and words falls below a given threshold (0.0001 in this study).

4 Empirical Evaluation

4.1 Summarization Evaluation

4.1.1 Evaluation Setup

We used task 1 of DUC2002 (DUC, 2002) for evaluation. The task aimed to evaluate generic summaries with a length of approximately 100 words or less. DUC2002 provided 567 English news articles collected from TREC-9 for single-document summarization task. The sentences in each article have been separated and the sentence information was stored into files.

In the experiments, the background corpus for using the mutual information measure to compute word semantics simply consisted of all the documents from DUC2001 to DUC2005, which could be easily expanded by adding more documents. The stopwords were removed and the remaining words were converted to the basic forms based on WordNet. Then the semantic similarity values between the words were computed.

We used the ROUGE (Lin and Hovy, 2003) toolkit (i.e.ROUGEeval-1.4.2 in this study) for evaluation, which has been widely adopted by DUC for automatic summarization evaluation. It measured summary quality by counting overlapping units such as the n-gram, word sequences and word pairs between the candidate summary and the reference summary. ROUGE toolkit reported separate scores for 1, 2, 3 and 4-gram, and also for longest common subsequence co-occurrences. Among these different scores, unigram-based ROUGE score (ROUGE-1) has been shown to agree with human judgment most (Lin and Hovy, 2003). We showed three of the ROUGE metrics in the experimental results: ROUGE-1 (unigram-based), ROUGE-2 (bigram-based), and ROUGE-W (based on weighted longest common subsequence, weight=1.2).

In order to truncate summaries longer than the length limit, we used the "-l" option [2] in the ROUGE toolkit.

4.1.2 Evaluation Results

For simplicity, the parameters in the proposed approach are simply set to $\alpha=\beta=0.5$, which means that the contributions from sentences and words are equally important. We adopt the WordNet-based *vector* measure (WN) and the corpus-based mutual information measure (MI) for computing the semantic similarity between words. When using the mutual information measure, we heuristically set the window size k to 2, 5 and 10, respectively.

The proposed approaches with different word similarity measures (WN and MI) are compared

[2] The "-l" option is very important for fair comparison. Some previous works not adopting this option are likely to overestimate the ROUGE scores.

with two solid baselines: SentenceRank and MutualRank. SentenceRank is proposed in Mihalcea and Tarau (2004) to make use of only the sentence-to-sentence relationships to rank sentences, which outperforms most popular summarization methods. MutualRank is proposed in Zha (2002) to make use of only the sentence-to-word relationships to rank sentences and words. For all the summarization methods, after the sentences are ranked by their saliency scores, we can apply a variant form of the MMR algorithm to remove redundancy and choose both the salient and novel sentences to the summary. Table 1 gives the comparison results of the methods before removing redundancy and Table 2 gives the comparison results of the methods after removing redundancy.

System	ROUGE-1	ROUGE-2	ROUGE-W
Our Approach (WN)	**0.47100**$^{*\#}$	**0.20424**$^{*\#}$	**0.16336**$^{\#}$
Our Approach (MI:k=2)	0.46711$^{\#}$	0.20195$^{\#}$	0.16257$^{\#}$
Our Approach (MI:k=5)	0.46803$^{\#}$	0.20259$^{\#}$	0.16310$^{\#}$
Our Approach (MI:k=10)	0.46823$^{\#}$	0.20301$^{\#}$	0.16294$^{\#}$
SentenceRank	0.45591	0.19201	0.15789
MutualRank	0.43743	0.17986	0.15333

Table 1. Summarization Performance before Removing Redundancy (w/o MMR)

System	ROUGE-1	ROUGE-2	ROUGE-W
Our Approach (WN)	**0.47329**$^{*\#}$	0.20249$^{\#}$	0.16352$^{\#}$
Our Approach (MI:k=2)	0.47281$^{\#}$	**0.20281**$^{\#}$	**0.16373**$^{\#}$
Our Approach (MI:k=5)	0.47282$^{\#}$	0.20249$^{\#}$	0.16343$^{\#}$
Our Approach (MI:k=10)	0.47223$^{\#}$	0.20225$^{\#}$	0.16308$^{\#}$
SentenceRank	0.46261	0.19457	0.16018
MutualRank	0.43805	0.17253	0.15221

Table 2. Summarization Performance after Removing Redundancy (w/ MMR)

(* indicates that the improvement over SentenceRank is significant and $^{\#}$ indicates that the improvement over MutualRank is significant, both by comparing the 95% confidence intervals provided by the ROUGE package.)

Seen from Tables 1 and 2, the proposed approaches always outperform the two baselines over all three metrics with different word semantic measures. Moreover, no matter whether the MMR algorithm is applied or not, almost all performance improvements over MutualRank are significant and the ROUGE-1 performance improvements over SentenceRank are significant when using WordNet-based measure (WN). Word semantics can be naturally incorporated into the computation process, which addresses the problem that SentenceRank cannot take into account word semantics, and thus improves the summarization performance. We also observe that the corpus-based measure (MI) works almost as well as the knowledge-based measure (WN) for computing word semantic similarity.

In order to better understand the relative contributions from the sentence nodes and the word nodes, the parameter α is varied from 0 to 1. The larger α is, the more contribution is given from the sentences through the SS-Graph, while the less contribution is given from the words through the SW-Graph. Figures 2-4 show the curves over three ROUGE scores with respect to α. Without loss of generality, we use the case of k=5 for the MI measure as an illustration. The curves are similar to Figures 2-4 when k=2 and k=10.

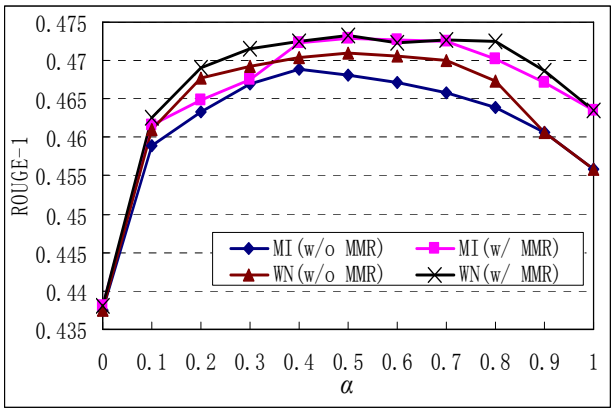


Figure 2. ROUGE-1 vs. α

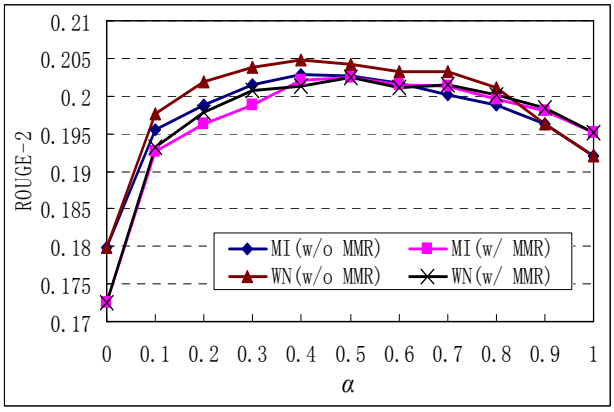


Figure 3. ROUGE-2 vs. α

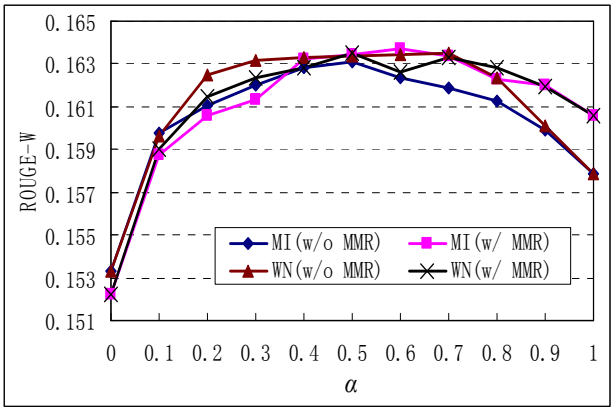


Figure 4. ROUGE-W vs. α

Seen from Figures 2-4, no matter whether the MMR algorithm is applied or not (i.e. w/o MMR or w/ MMR), the ROUGE scores based on either word semantic measure (MI or WN) achieves the peak when α is set between 0.4 and 0.6. The performance values decrease sharply when α is very large (near to 1) or very small (near to 0). The curves demonstrate that both the contribution from the sentences and the contribution from the words are important for ranking sentences; moreover, the contributions are almost equally important. Loss of either contribution will much deteriorate the final performance.

Similar results and observations have been obtained on task 1 of DUC2001 in our study and the details are omitted due to page limit.

4.2 Keyword Evaluation

4.1.1 Evaluation Setup

In this study we performed a preliminary evaluation of keyword extraction. The evaluation was conducted on the single word level instead of the multi-word phrase (n-gram) level, in other words, we compared the automatically extracted unigrams (words) and the manually labeled unigrams (words). The reasons were that: 1) there existed partial matching between phrases and it was not trivial to define an accurate measure to evaluate phrase quality; 2) each phrase was in fact composed of a few words, so the keyphrases could be obtained by combining the consecutive keywords.

We used 34 documents in the first five document clusters in DUC2002 dataset (i.e. d061-d065). At most 10 salient words were manually labeled for each document to represent the document and the average number of manually assigned keywords was 6.8. Each approach returned 10 words with highest saliency scores as the keywords. The extracted 10 words were compared with the manually labeled keywords. The words were converted to their corresponding basic forms based on WordNet before comparison. The precision p, recall r, F-measure (F=2pr/(p+r)) were obtained for each document and then the values were averaged over all documents for evaluation purpose.

4.1.2 Evaluation Results

Table 3 gives the comparison results. The proposed approaches are compared with two baselines: WordRank and MutualRank. WordRank is proposed in Mihalcea and Tarau (2004) to make use of only the co-occurrence relationships between words to rank words, which outperforms traditional keyword extraction methods. The window size k for WordRank is also set to 2, 5 and 10, respectively.

System	Precision	Recall	F-measure
Our Approach (WN)	0.413	**0.504**	0.454
Our Approach (MI:k=2)	**0.428**	0.485	0.455
Our Approach (MI:k=5)	0.425	0.491	**0.456**
Our Approach (MI:k=10)	0.393	0.455	0.422
WordRank (k=2)	0.373	0.412	0.392
WordRank (k=5)	0.368	0.422	0.393
WordRank (k=10)	0.379	0.407	0.393
MutualRank	0.355	0.397	0.375

Table 3. The Performance of Keyword Extraction

Seen from the table, the proposed approaches significantly outperform the baseline approaches. Both the corpus-based measure (MI) and the knowledge-based measure (WN) perform well on the task of keyword extraction.

A running example is given below to demonstrate the results:

Document ID: *D062/AP891018-0301*

Labeled keywords:

insurance earthquake insurer damage california Francisco pay

Extracted keywords:

WN: *insurance earthquake insurer quake california spokesman cost million wednesday damage*

MI(k=5): *insurance insurer earthquake percent benefit california property damage estimate rate*

5 Conclusion and Future Work

In this paper we propose a novel approach to simultaneously document summarization and keyword extraction for single documents by fusing the sentence-to-sentence, word-to-word, sentence-to-word relationships in a unified framework. The semantics between words computed by either corpus-based approach or knowledge-based approach can be incorporated into the framework in a natural way. Evaluation results demonstrate the performance improvement of the proposed approach over the baselines for both tasks.

In this study, only the mutual information measure and the *vector* measure are employed to compute word semantics, and in future work many other measures mentioned earlier will be investigated in the framework in order to show the robustness of the framework. The evaluation of keyword extraction is preliminary in this study, and we will conduct more thorough experiments to make the results more convincing. Furthermore, the proposed approach will be applied to multi-document summarization and keyword extraction, which are considered more difficult than single document summarization and keyword extraction.

Acknowledgements

This work was supported by the National Science Foundation of China (60642001).

References

M. R. Amini and P. Gallinari. 2002. The use of unlabeled data to improve supervised learning for text summarization. In *Proceedings of SIGIR2002*, 105-112.

S. Brin and L. Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7).

J. Carbonell and J. Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of SIGIR-1998*, 335-336.

J. M. Conroy and D. P. O'Leary. 2001. Text summarization via Hidden Markov Models. In *Proceedings of SIGIR2001*, 406-407.

DUC. 2002. The Document Understanding Workshop 2002. http://www-nlpir.nist.gov/projects/duc/guidelines/2002.html

T. Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics* 19, 61–74.

G. ErKan and D. R. Radev. 2004. LexPageRank: Prestige in multi-document text summarization. In *Proceedings of EMNLP2004*.

C. Fellbaum. 1998. WordNet: An Electronic Lexical Database. The MIT Press.

E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. 1999. Domain-specific keyphrase extraction. *Proceedings of IJCAI-99*, pp. 668-673.

Y. H. Gong and X. Liu. 2001. Generic text summarization using Relevance Measure and Latent Semantic Analysis. In *Proceedings of SIGIR2001*, 19-25.

E. Hovy and C. Y. Lin. 1997. Automated text summarization in SUMMARIST. In *Proceeding of ACL'1997/EACL'1997 Worshop on Intelligent Scalable Text Summarization*.

A. Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of EMNLP2003*, Japan, August.

J. M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.

B. Krulwich and C. Burkey. 1996. Learning user information interests through the extraction of semantically significant phrases. In *AAAI 1996 Spring Symposium on Machine Learning in Information Access*.

J. Kupiec, J. Pedersen, and F. Chen. 1995. A.trainable document summarizer. In Proceedings of SIGIR1995, 68-73.

T. K. Landauer, P. Foltz, and D. Laham. 1998. Introduction to latent semantic analysis. *Discourse Processes* 25.

C. Y. Lin and E. Hovy. 2000. The automated acquisition of topic signatures for text Summarization. In *Proceedings of ACL-2000*, 495-501.

C.Y. Lin and E.H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of HLT-NAACL2003*, Edmonton, Canada, May.

R. Mihalcea, C. Corley, and C. Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of AAAI-06*.

R. Mihalcea and P. Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of EMNLP2004*.

R. Mihalcea and P.Tarau. 2005. A language independent algorithm for single and multiple document summarization. In *Proceedings of IJCNLP2005*.

A. Muñoz. 1996. Compound key word generation from document databases using a hierarchical clustering ART model. *Intelligent Data Analysis,* 1(1).

T. Nomoto and Y. Matsumoto. 2001. A new approach to unsupervised text summarization. In *Proceedings of SIGIR2001*, 26-34.

S. Patwardhan. 2003. Incorporating dictionary and corpus information into a context vector measure of semantic relatedness. *Master's thesis*, Univ. of Minnesota, Duluth.

T. Pedersen, S. Patwardhan, and J. Michelizzi. 2004. WordNet::Similarity – Measuring the relatedness of concepts. In *Proceedings of AAAI-04*.

D. Shen, J.-T. Sun, H. Li, Q. Yang, and Z. Chen. 2007. Document Summarization using Conditional Random Fields. In *Proceedings of IJCAI 07*.

A. M. Steier and R. K. Belew. 1993. Exporting phrases: A statistical analysis of topical language. In *Proceedings of* S*econd Symposium on Document Analysis and Information Retrieval*, pp. 179-190.

P. D. Turney. 2000. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2:303-336.

P. Turney. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of ECML-2001*.

I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. 1999. KEA: Practical automatic keyphrase extraction. *Proceedings of Digital Libraries 99 (DL'99),* pp. 254-256.

H. Y. Zha. 2002. Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In *Proceedings of SIGIR2002*, pp. 113-120.

Fast Semantic Extraction Using a Novel Neural Network Architecture

Ronan Collobert
NEC Laboratories America, Inc.
4 Independence Way
Suite 200, Princeton, NJ 08540
collober@nec-labs.com

Jason Weston
NEC Laboratories America, Inc.
4 Independence Way
Suite 200, Princeton, NJ 08540
jasonw@nec-labs.com

Abstract

We describe a novel neural network architecture for the problem of *semantic role labeling*. Many current solutions are complicated, consist of several stages and hand-built features, and are too slow to be applied as part of real applications that require such semantic labels, partly because of their use of a syntactic parser (Pradhan et al., 2004; Gildea and Jurafsky, 2002). Our method instead learns a direct mapping from source sentence to semantic tags for a given predicate without the aid of a parser *or* a chunker. Our resulting system obtains accuracies comparable to the current state-of-the-art at a fraction of the computational cost.

1 Introduction

Semantic understanding plays an important role in many end-user applications involving text: for information extraction, web-crawling systems, question and answer based systems, as well as machine translation, summarization and search. Such applications typically have to be *computationally cheap* to deal with an enormous quantity of data, e.g. web-based systems process large numbers of documents, whilst interactive human-machine applications require almost instant response. Another issue is the cost of producing labeled training data required for statistical models, which is exacerbated when those models also depend on syntactic features which must themselves be learnt.

To achieve the goal of semantic understanding, the current consensus is to divide and conquer the problem. Researchers tackle several layers of processing tasks ranging from the syntactic, such as part-of-speech labeling and parsing, to the semantic: word-sense disambiguation, semantic role-labeling, named entity extraction, co-reference resolution and entailment. None of these tasks are end goals in themselves but can be seen as layers of feature extraction that can help in a language-based end application, such as the ones described above. Unfortunately, the state-of-the-art solutions of many of these tasks are simply too slow to be used in the applications previously described. For example, state-of-the-art syntactic parsers theoretically have cubic complexity in the sentence length (Younger, 1967)[1] and several semantic extraction algorithms use the parse tree as an initial feature.

[The company]$_{\text{ARG0}}$ [bought]$_{\text{REL}}$ [sugar]$_{\text{ARG1}}$ [on the world market]$_{\text{ARGM-LOC}}$ [to meet export commitments]$_{\text{ARGM-PNC}}$

Figure 1: Example of *Semantic Role Labeling* from the PropBank dataset (Palmer et al., 2005). ARG0 is typically an actor, REL an action, ARG1 an object, and ARGM describe various modifiers such as location (LOC) and purpose (PNC).

In this work, we describe a novel type of neural network architecture that could help to solve some of these issues. We focus our experimental study on the *semantic role labeling* problem (Palmer et al., 2005): being able to give a semantic role to a syn-

[1] Even though some parsers effectively exhibit linear behavior in sentence length (Ratnaparkhi, 1997), fast statistical parsers such as (Henderson, 2004) still take around 1.5 seconds for sentences of length 35 in tests that we made.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 560–567,
Prague, Czech Republic, June 2007.

tactic constituent of a sentence, i.e. annotating the predicate argument structure in text (see for example Figure 1). Because of its nature, role labeling seems to require the syntactic analysis of a sentence before attributing semantic labels. Using this intuition, state-of-the-art systems first build a *parse tree*, and syntactic constituents are then labeled by feeding hand-built features extracted from the parse tree to a machine learning system, e.g. the ASSERT system (Pradhan et al., 2004). This is rather slow, taking a few seconds per sentence at test time, partly because of the parse tree component, and partly because of the use of Support Vector Machines (Boser et al., 1992), which have linear complexity in testing time with respect to the number of training examples. This makes it hard to apply this method to interesting end user applications.

Here, we propose a radically different approach that avoids the more complex task of building a full parse tree. From a machine learning point of view, a human does not need to be taught about parse trees to talk. It is possible, however, that our brains may implicitly learn features highly correlated with those extracted from a parse tree. We propose to develop an architecture that implements this kind of implicit learning, rather than using explicitly engineered features. In practice, our system also provides semantic tags at a fraction of the computational cost of other methods, taking on average 0.02 seconds to label a sentence from the Penn Treebank, with almost no loss in accuracy.

The rest of the article is as follows. First, we describe the problem of shallow semantic parsing in more detail, as well as existing solutions to this problem. We then detail our algorithmic approach – the neural network architecture we employ – followed by experiments that evaluate our method. Finally, we conclude with a summary and discussion of future work.

2 Shallow Semantic Parsing

FrameNet (Baker et al., 1998) and the Proposition Bank (Palmer et al., 2005), or PropBank for short, are the two main systems currently developed for semantic role-labeling annotation. We focus here on PropBank. PropBank encodes role labels by semantically tagging the syntactic structures of hand annotated parses of sentences. The current version of the dataset gives semantic tags for the same sentences as in the Penn Treebank (Marcus et al., 1993), which are excerpts from the Wall Street Journal. The central idea is that each verb in a sentence is labeled with its propositional arguments, where the abstract numbered arguments are intended to fill typical roles. For example, ARG0 is typically the actor, and ARG1 is typically the thing acted upon. The precise usage of the numbering system is labeled for each particular verb as so-called frames. Additionally, semantic roles can also be labeled with one of 13 ARGM adjunct labels, such as ARGM-LOC or ARGM-TMP for additional locational or temporal information relative to some verb.

Shallow semantic parsing has immediate applications in tasks such as meta-data extraction (e.g. from web documents) and question and answer based systems (e.g. call center systems), amongst others.

3 Previous Work

Several authors have already attempted to build machine learning approaches for the semantic role-labeling problem. In (Gildea and Jurafsky, 2002) the authors presented a statistical approach to learning (for FrameNet), with some success. They proposed to take advantage of the syntactic tree structure that can be predicted by a parser, such as Charniak's parser (Charniak, 2000). Their aim is, given a node in the parse tree, to assign a semantic role label to the words that are the children of that node. They extract several key types of features from the parse tree to be used in a statistical model for prediction. These same features also proved crucial to subsequent approaches, e.g. (Pradhan et al., 2004). These features include:

- The parts of speech and syntactic labels of words and nodes in the tree.
- The node's position (left or right) in relation to the verb.
- The syntactic path to the verb in the parse tree.
- Whether a node in the parse tree is part of a noun or verb phrase (by looking at the parent nodes of that node).

- The voice of the sentence: active or passive (part of the PropBank gold annotation);

as well as several other features (predicate, head word, verb sub-categorization, ...).

The authors of (Pradhan et al., 2004) used a similar structure, but added more features, notably head word part-of-speech, the predicted named entity class of the argument, word sense disambiguation of the verb and verb clustering, and others (they add 25 variants of 12 new feature types overall.) Their system also uses a parser, as before, and then a polynomial Support Vector Machine (SVM) (Boser et al., 1992) is used in two further stages: to classify each node in the tree as being a semantic argument or not for a given verb; and then to classify each semantic argument into one of the classes (ARG1, ARG2, etc.). The first SVM solves a two-class problem, the second solves a multi-class problem using a one-vs-the-rest approach. The final system, called ASSERT, gives state-of-the-art performance and is also freely available at: `http://oak.colorado.edu/assert/`. We compare to this system in our experimental results in Section 5. Several other competing methods exist, e.g. the ones that participated in the CONLL 2004 and 2005 challenges (`http://www.lsi.upc.edu/~srlconll/st05/st05.html`). In this paper we focus on a comparison with ASSERT because software to re-run it is available online. This also gives us a timing result for comparison purposes.

The three-step procedure used in ASSERT (calculating a parse tree and then applying SVMs twice) leads to good classification performance, but has several drawbacks. First in speed: predicting a parse tree is extremely demanding in computing resources. Secondly, choosing the features necessary for SVM classification requires extensive research. Finally, the SVM classification algorithm used in existing approaches is rather slow: SVM training is at least quadratic in time with respect to the number of training examples. The number of support vectors involved in the SVM decision function also increases linearly with the number of training examples. This makes SVMs slow on large-scale problems, both during training and testing phases.

To alleviate the burden of parse tree computation, several attempts have been made to remove the full parse tree information from the semantic role labeling system, in fact the shared task of CONLL 2004 was devoted to this goal, but the results were not completely satisfactory. Previously, in (Gildea and Palmer, 2001), the authors tried to show that the parse tree is necessary for good generalization by showing that segments derived from a shallow syntactic parser *or* chunker do not perform as well for this goal. A further analysis of using chunkers, with improved results was also given in (Punyakanok et al., 2005), but still concluded the full parse tree is most useful.

4 Neural Network Architecture

Ideally, we want an end-to-end *fast* learning system to output semantic roles for syntactic constituents without using a time consuming parse tree.

Also, as explained before, we are interesting in exploring whether machine learning approaches can learn structure implicitly. Hence, even if there is a deep relationship between syntax and semantics, we prefer to avoid hand-engineered features that exploit this, and see if we can develop a model that can learn these features instead. We are thus not interested in chunker-based techniques, even though they are faster than parser-based techniques.

We propose here a *neural network based architecture* which achieves these two goals.

4.1 Basic Architecture

The type of neural network that we employ is a Multi Layer Perceptron (MLP). MLPs have been used for many years in the machine learning field and slowly abandoned for several reasons: partly because of the difficulty of solving the non-convex optimization problems associated with learning (LeCun et al., 1998), and partly because of the difficulty of their theoretical analysis compared to alternative convex approaches.

An MLP works by successively projecting the data to be classified into different spaces. These projections are done in what is called *hidden layers*. Given an input vector $\boldsymbol{z}$, a hidden layer applies a linear transformation (matrix M) followed by a squashing function h:

$$\boldsymbol{z} \mapsto M\boldsymbol{z} \mapsto h(M\boldsymbol{z})\,. \qquad (1)$$

A typical squashing function is the hyperbolic tangent $h(\cdot) = \tanh(\cdot)$. The last layer (the output layer) linearly separates the classes. The composition of the projections in the hidden layers could be viewed as the work done by the kernel in SVMs. However there is a very important difference: the kernel in SVM is fixed and arbitrarily chosen, while the hidden layers in an MLP are *trained* and *adapted* to the classification task. This allows us to create much more flexible classification architectures.

Our method for semantic role labeling classifies each word of a sentence separately. We do not use any semantic constituent information: if the model is powerful enough, words in the same semantic constituent should have the same class label. This means we also do not separate the problem into an identification and classification phase, but rather solve in a single step.

4.1.1 Notation

We represent words as indices. We consider a finite dictionary of words $\mathcal{D} \subset \mathbb{N}$. Let us represent a sentence of n_w words to be analyzed as a function $s(\cdot)$. The i^{th} word in the sentence is given by the index $s(i)$:

$$1 \leq i \leq n_w \quad s(i) \in \mathcal{D}\,.$$

We are interested in predicting the semantic role label of the word at position pos_w, given a verb at position pos_v ($1 \leq pos_w, pos_v \leq n_w$). A mathematical description of our network architecture schematically shown in Figure 2 follows.

4.1.2 Transforming words into feature vectors

Our first concern in semantic role labeling is that we have to deal with words, and that a simple index $i \in \mathcal{D}$ does not carry any information specific to a word: for each word we need a set of features relevant for the task. As described earlier, previous methods construct a parse tree, and then compute hand-built features which are then fed to a classification algorithm. In order to bypass the use of a parse tree, we convert each word $i \in \mathcal{D}$ into a particular vector $\boldsymbol{w}_i \in \mathbb{R}^d$ *which is going to be learnt* for the task we are interested in. This approach has already been used with great success in the domain of language models (Bengio and Ducharme, 2001; Schwenk and Gauvain, 2002).

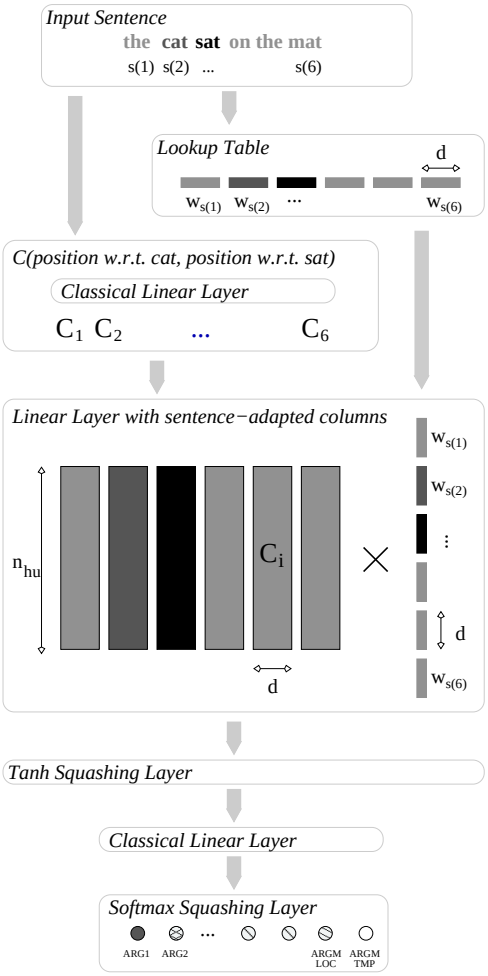


Figure 2: MLP architecture for shallow semantic parsing. The input sequence is at the top. The output class probabilities for the word of interest ("cat") given the verb of interest ("sat") are given at the bottom.

The first layer of our MLP is thus a lookup table which replaces the word indices into a concatenation of vectors:

$$\begin{aligned}\{s(1), \ldots, s(n_w)\} \\ \mapsto (\boldsymbol{w}_{s(1)} \ \ldots \ \boldsymbol{w}_{s(n_w)}) \in \mathbb{R}^{n_w\, d}\,.\end{aligned} \tag{2}$$

The weights $\{\boldsymbol{w}_i \,|\, i \in \mathcal{D}\}$ for this layer are considered during the backpropagation phase of the MLP, and thus adapted automatically for the task we are interested in.

4.1.3 Integrating the verb position

Feeding word vectors alone to a linear classification layer as in (Bengio and Ducharme, 2001) leads

to very poor accuracy because the semantic classification of a given word also depends on the verb in question. We need to provide the MLP with information about the verb position within the sentence. For that purpose we use a kind of linear layer which is *adapted* to the sentence considered. It takes the form:

$$(\boldsymbol{w}_{s(1)} \ \dots \ \boldsymbol{w}_{s(n_w)}) \mapsto M \begin{pmatrix} \boldsymbol{w}_{s(1)}^{\mathrm{T}} \\ \vdots \\ \boldsymbol{w}_{s(n_w)}^{\mathrm{T}} \end{pmatrix} ,$$

where $M \in \mathbb{R}^{n_{hu} \times n_w \, d}$, and n_{hu} is the number of hidden units. The specific nature of this layer is that the matrix M has a special block-column form which depends on the sentence:

$$M = (C_1 | \dots | C_{n_w}) ,$$

where each column $C_i \in \mathbb{R}^{n_{hu} \times d}$ depends on the position of the i^{th} word in $s(\cdot)$, with respect to the position pos_w of the word of interest, and with respect to the position pos_v of the verb of interest:

$$C_i = \mathcal{C}(i - pos_w, \, i - pos_v) ,$$

where $\mathcal{C}(\cdot, \cdot)$ is a function to be chosen.

In our experiments $\mathcal{C}(\cdot, \cdot)$ was a linear layer with discretized inputs $(i - pos_w, \, i - pos_v)$ which were transformed into two binary vectors of size wsz, where a bit is set to 1 if it corresponds to the position to encode, and 0 otherwise. These two binary vectors are then concatenated and fed to the linear layer. We chose the "window size" $wsz = 11$. If a position lies outside the window, then we still set the leftmost or rightmost bit to 1. The parameters involved in this function are also considered during the backpropagation. With such an architecture we allow our MLP to automatically adapt the importance of a word in the sentence given its distance to the word we want to classify, and to the verb we are interested in.

This idea is the major novelty in this work, and is crucial for the success of the entire architecture, as we will see in the experiments.

4.1.4 Learning class probabilities

The last layer in our MLP is a classical linear layer as described in (1), with a *softmax* squashing function (Bridle, 1990). Considering (1) and given $\tilde{\boldsymbol{z}} = M\boldsymbol{z}$, we have

$$h_i(\tilde{\boldsymbol{z}}) = \frac{\exp \tilde{z}_i}{\sum_j \exp \tilde{z}_j} .$$

This allows us to interpret outputs as probabilities for each semantic role label. The training of the whole system is achieved using a normal stochastic gradient descent.

4.2 Word representation

As we have seen, in our model we are *learning* one d dimensional vector to represent each word. If the dataset were large enough, this would be an elegant solution. In practice many words occur infrequently within PropBank, so (independent of the size of d) we can still only learn a very poor representation for words that only appear a few times. Hence, to control the capacity of our model we take the original word and replace it with its part-of-speech if it is a verb, noun, adjective, adverb or number as determined by a part-of-speech classifier, and keep the words for all other parts of speech. This classifier is itself a neural network. This way we keep linking words which are important for this task. We do not do this replacement for the predicate itself.

5 Experiments

We used Sections 02-21 of the PropBank dataset version 1 for training and validation and Section 23 for testing as standard in all our experiments. We first describe the part-of-speech tagger we employ, and then describe our semantic role labeling experiments. Software for our method, SENNA (Semantic Extraction using a Neural Network Architecture), more details on its implementation, an online applet and test set predictions of our system in comparison to ASSERT can be found at `http://ml.nec-labs.com/software/senna`.

Part-Of-Speech Tagger The part-of-speech classifier we employ is a neural network architecture of the same type as in Section 4, where the function $C_i = C(i - pos_w)$ depends now only on the word position, and not on a verb. More precisely:

$$C_i = \begin{cases} 0 & \text{if } 2\,|i - pos_w| > wsz - 1 \\ W_{i - pos_w} & \text{otherwise} , \end{cases}$$

where $W_k \in \mathbb{R}^{n_{hu} \times d}$ and wsz is a window size. We chose $wsz = 5$ in our experiments. The d-dimensional vectors learnt take into account the capitalization of a word, and the prefix and suffix calculated using Porter-Stemmer. See `http://ml.nec-labs.com/software/senna` for more details. We trained on the training set of PropBank supplemented with the Brown corpus, resulting in a test accuracy on the test set of PropBank of 96.85% which compares to 96.66% using the Brill tagger (Brill, 1992).

Semantic Role Labeling In our experiments we considered a 23-class problem of NULL (no label), the core arguments ARG0-5, REL, ARGA, and ARGM- along with the 13 secondary modifier labels such as ARGM-LOC and ARGM-TMP. We simplified R-ARGn and C-ARGn to be written as ARGn, and post-processed ASSERT to do this as well.

We compared our system to the freely available ASSERT system (Pradhan et al., 2004). Both systems are fed only the input sentence during testing, with traces removed, so they cannot make use of many PropBank features such as frameset identitifier, person, tense, aspect, voice, and form of the verb. As our algorithm outputs a semantic tag for each word of a sentence, we directly compare this per-word accuracy with ASSERT. Because ASSERT uses a parser, and because PropBank was built by labeling the nodes of a hand-annotated parse tree, per-node accuracy is usually reported in papers such as (Pradhan et al., 2004). Unfortunately our approach is based on a completely different premise: we tag words, not syntactic constituents coming from the parser. We discuss this further in Section 5.2.

The per-word accuracy comparison results can be seen in Table 5. Before labeling the semantic roles of each predicate, one must first identify the predicates themselves. If a predicate is not identified, NULL tags are assigned to each word for that predicate. The first line of results in the table takes into account this identification process. For the neural network, we used our part-of-speech tagger to perform this as a verb-detection task.

We noticed ASSERT failed to identify relatively many predicates. In particular, it seems predicates such as "is" are sometimes labeled as AUX by the part-of-speech tagger, and subsequently ignored. We informed the authors of this, but we did not receive a response. To deal with this, we considered the additional accuracy (second row in the table) measured over only those sentences where the predicate was identified by ASSERT.

Timing results The per-sentence compute time is also given in Table 5, averaged over all sentences in the test set. Our method is around 250 times faster than ASSERT. It is not really feasible to run ASSERT for most applications.

Measurement	NN	ASSERT
Per-word accuracy (all verbs)	83.64%	83.46%
Per-word accuracy (ASSERT verbs)	84.09%	86.06%
Per-sentence compute time (secs)	0.02 secs	5.08 secs

Table 1: Experimental comparison with ASSERT

5.1 Analysis of our MLP

While we gave an intuitive justification of the architecture choices of our model in Section 4, we now give a systematic empirical study of those choices. First of all, providing the position of the word *and* the predicate in function $C(\cdot, \cdot)$ is essential: the best model we obtained with a window around the word only gave 51.3%, assuming correct identification of all predicates. Our best model achieves 83.95% in this setting.

If we do not cluster the words according to their part-of-speech, we also lose some performance, obtaining 78.6% at best. On the other hand, clustering *all* words (such as CC, DT, IN part-of-speech tags) also gives weaker results (81.1% accuracy at best). We believe that including all words would give very good performance if the dataset was large enough, but training only on PropBank leads to overfitting, many words being infrequent. Clustering is a way to fight against overfitting, by grouping infrequent words: for example, words with the label NNP, JJ, RB (which we cluster) appear on average 23, 22 and 72 times respectively in the training set, while CC, DT, IN (which we do not cluster) appear 2420, 5659 and 1712 times respectively.

Even though some verbs are infrequent, one cannot cluster all verbs into a single group, as each verb dictates the types of semantic roles in the sentence, depending on its frame. Clustering all words into their part-of-speech, including the predicate, gives a poor 73.8% compared with 81.1%, where everything is clustered apart from the predicate.

Figure 3 gives some anecdotal examples of test set predictions of our final model compared to ASSERT.

5.2 Argument Classification Accuracy

So far we have not used the same accuracy measures as in previous work (Gildea and Jurafsky, 2002; Pradhan et al., 2004). Currently our architecture is designed to label on a per-word basis, while existing systems perform a segmentation process, and then label segments. While we do not optimize our model for the same criteria, it is still possible to measure the accuracy using the same metrics. We measured the argument classification accuracy of our network, assuming the correct segmentation is given to our system, as in (Pradhan et al., 2004), by post-processing our per-word tags to form a majority vote over each segment. This gives 83.18% accuracy for our network when we suppose the predicate must also be identified, and 80.53% for the ASSERT software. Measuring only on predicates identified by ASSERT we instead obtain 84.32% accuracy for our network, and 87.02% for ASSERT.

TRUTH: He camped out at a high-tech nerve center on the floor of [the Big Board, where]$_{\text{ARGM-LOC}}$ [he]$_{\text{ARG0}}$ [could]$_{\text{ARGM-MOD}}$ [watch]$_{\text{REL}}$ [updates on prices and pending stock orders]$_{\text{ARG1}}$.
ASSERT (68.7%): He camped out at a high-tech nerve center on the floor of the Big Board, [where]$_{\text{ARGM-LOC}}$ [he]$_{\text{ARG0}}$ [could]$_{\text{ARGM-MOD}}$ [watch]$_{\text{REL}}$ [updates]$_{\text{ARG1}}$ on prices and pending stock orders.
NN (100%): He camped out at a high-tech nerve center on the floor of [the Big Board, where]$_{\text{ARGM-LOC}}$ [he]$_{\text{ARG0}}$ [could]$_{\text{ARGM-MOD}}$ [watch]$_{\text{REL}}$ [updates on prices and pending stock orders]$_{\text{ARG1}}$.

TRUTH: [United Auto Workers Local 1069, which]$_{\text{ARG0}}$ [represents]$_{\text{REL}}$ [3,000 workers at Boeing's helicopter unit in Delaware County, Pa.]$_{\text{ARG1}}$, said it agreed to extend its contract on a day-by-day basis, with a 10-day notification to cancel, while it continues bargaining.
ASSERT (100%): [United Auto Workers Local 1069, which]$_{\text{ARG0}}$ [represents]$_{\text{REL}}$ [3,000 workers at Boeing's helicopter unit in Delaware County, Pa.]$_{\text{ARG1}}$, said it agreed to extend its contract on a day-by-day basis, with a 10-day notification to cancel, while it continues bargaining.
NN (89.1%): [United Auto Workers Local 1069, which]$_{\text{ARG0}}$ [represents]$_{\text{REL}}$ [3,000 workers at Boeing's helicopter unit]$_{\text{ARG1}}$ [in Delaware County]$_{\text{ARGM-LOC}}$, Pa. , said it agreed to extend its contract on a day-by-day basis, with a 10-day notification to cancel, while it continues bargaining.

Figure 3: Two examples from the PropBank test set, showing Neural Net and ASSERT and gold standard labelings, with per-word accuracy in brackets. Note that even though our labeling does not match the hand-annotated one in the second sentence it still seems to make some sense as "in Delaware County" is labeled as a location modifier. The complete set of predictions on the test set can be found at `http://ml.nec-labs.com/software/senna`.

6 Discussion

We have introduced a neural network architecture that can provide computationally efficient semantic role tagging. It is also a general architecture that could be applied to other problems as well. Because our network currently outputs labels on a per-word basis it is difficult to assess existing accuracy measures. However, it should be possible to combine our approach with a shallow parser to enhance performance, and make comparisons more direct.

We consider this work as a starting point for different research directions, including the following areas:

- *Incorporating hand-built features* Currently, the only prior knowledge our system encodes comes from part-of-speech tags, in stark contrast to other methods. Of course, performance would improve with more hand-built features. For example, simply adding whether each word is part of a noun or verb phrase using the hand-annotated parse tree (the so-called "GOV" feature from (Gildea and Jurafsky, 2002)) improves the performance of our system from 83.95% to 85.8%. One must trade the generality of the model with its specificity, and also take into account how long the features take to compute.

- *Incorporating segment information* Our system has no prior knowledge about segmentation in text. This could be encoded in many ways: most obviously by using a chunker, but also by

designing a different network architecture, e.g. by encoding contiguity constraints. To show the latter is useful, using hand-annotated segments to force contiguity by majority vote leads to an improvement from 83.95% to 85.6%.

- *Incorporating known invariances via virtual training data.* In image recognition problems it is common to create artificial training data by taking into account invariances in the images, e.g. via rotation and scale. Such data improves generalization substantially. It may be possible to achieve similar results for text, by "warping" training data to create new sentences, or by constructing sentences from scratch using a hand-built grammar.

- *Unlabeled data.* Our representation of words is as d dimensional vectors. We could try to improve this representation by learning a language model from unlabeled data (Bengio and Ducharme, 2001). As many words in PropBank only appear a few times, the representation might improve, even though the learning is unsupervised. This may also make the system generalize better to types of data other than the Wall Street Journal.

- *Transductive Inference.* Finally, one can also use unlabeled data as part of the supervised training process, which is called transduction or semi-supervised learning.

In particular, we find the possibility of using unlabeled data, invariances and the use of transduction exciting. These possibilities naturally fit into our framework, whereas scalability issues will limit their application in competing methods.

References

C.F. Baker, C.J. Fillmore, and J.B. Lowe. 1998. The Berkeley FrameNet project. *Proceedings of COLING-ACL*, 98.

Y. Bengio and R. Ducharme. 2001. A neural probabilistic language model. In *Advances in Neural Information Processing Systems, NIPS 13*.

B.E. Boser, I.M. Guyon, and V.N. Vapnik. 1992. A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152.

J.S. Bridle. 1990. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In F. Fogelman Soulié and J. Hérault, editors, *Neurocomputing: Algorithms, Architectures and Applications*, pages 227–236. NATO ASI Series.

E. Brill. 1992. A simple rule-based part of speech tagger. *Proceedings of the Third Conference on Applied Natural Language Processing*, pages 152–155.

E. Charniak. 2000. A maximum-entropy-inspired parser. *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, pages 132–139.

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

D. Gildea and M. Palmer. 2001. The necessity of parsing for predicate argument recognition. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 239–246.

J. Henderson. 2004. Discriminative training of a neural network statistical parser. In *Proceedings of the 42nd Meeting of Association for Computational Linguistics.*

Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. 1998. Efficient backprop. In G.B. Orr and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, pages 9–50. Springer.

M.P. Marcus, M.A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of English: the penn treebank. *Computational Linguistics*, 19(2):313–330.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Comput. Linguist.*, 31(1):71–106.

S. Pradhan, W. Ward, K. Hacioglu, J. Martin, and D. Jurafsky. 2004. Shallow semantic parsing using support vector machines. *Proceedings of HLT/NAACL-2004*.

V. Punyakanok, D. Roth, and W. Yih. 2005. The necessity of syntactic parsing for semantic role labeling. *Proceedings of IJCAI'05*, pages 1117–1123.

A. Ratnaparkhi. 1997. A linear observed time statistical parser based on maximum entropy models. *Proceedings of EMNLP*.

H. Schwenk and J.L. Gauvain. 2002. Connectionist language modeling for large vocabulary continuousspeech recognition. *Proceedings of ICASSP'02*.

D. H. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10.

Improving the Interpretation of Noun Phrases with Cross-linguistic Information

Roxana Girju
University of Illinois at Urbana-Champaign
girju@uiuc.edu

Abstract

This paper addresses the automatic classification of semantic relations in noun phrases based on cross-linguistic evidence from a set of five Romance languages. A set of novel semantic and contextual English–Romance NP features is derived based on empirical observations on the distribution of the syntax and meaning of noun phrases on two corpora of different genre (Europarl and CLUVI). The features were employed in a Support Vector Machines algorithm which achieved an accuracy of 77.9% (Europarl) and 74.31% (CLUVI), an improvement compared with two state-of-the-art models reported in the literature.

1 Introduction

Semantic knowledge is very important for any application that requires a deep understanding of natural language. The automatic acquisition of semantic information in text has become increasingly important in ontology development, information extraction, question answering, and other advanced natural language processing applications.

In this paper we present a model for the automatic semantic interpretation of noun phrases (NPs), which is the task of determining the semantic relation among the noun constituents. For example, *family estate* encodes a POSSESSION relation, while *dress of silk* refers to PART-WHOLE. The problem, while simple to state is hard to solve. The reason is that the meaning of these constructions is most of the time ambiguous or implicit. Interpreting NPs correctly requires various types of information from world knowledge to complex context features. Moreover, the extension of this task to other natural languages brings forward new issues and problems. For instance, *beer glass* translates into *tarro de cerveza* in Spanish, *bicchiere da birra* in Italian, *verre à bière* in French, and *pahar de bere* in Romanian. Thus, an important research question is how do the syntactic constructions in the target language contribute to the preservation of meaning in context.

In this paper we investigate noun phrases based on cross-linguistic evidence and present a domain independent model for their semantic interpretation. We aim at uncovering the general aspects that govern the semantics of NPs in English based on a set of five Romance languages: Spanish, Italian, French, Portuguese, and Romanian. The focus on Romance languages is well motivated. It is mostly true that English noun phrases translate into constructions of the form *N P N* in Romance languages where, as we will show below, the *P* (preposition) varies in ways that correlate with the semantics. Thus Romance languages will give us another source of evidence for disambiguating the semantic relations in English NPs. We also present empirical observations on the distribution of the syntax and meaning of noun phrases on two different corpora based on two state-of-the-art classification tag sets: Lauer's set of 8 prepositions (Lauer, 1995) and our list of 22 semantic relations. We show that various crosslingual cues can help in the NP interpretation task when employed in an SVM model. The results are compared against two state of the art approaches: a su-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 568–575,
Prague, Czech Republic, June 2007.

pervised machine learning model, Semantic Scattering (Moldovan and Badulescu, 2005), and a web-based probabilistic model (Lapata and Keller, 2004).

The paper is organized as follows. In Section 2 we present a summary of the previous work. Section 3 lists the syntactic and semantic interpretation categories used along with observations regarding their distribution on the two different cross-lingual corpora. Sections 4 and 5 present a learning model and results for the interpretation of English noun phrases. Finally, in Section 6 we offer some discussion and conclusions.

2 Related Work

Currently, the best-performing NP interpretation methods in computational linguistics focus mostly on two consecutive noun instances (noun compounds) and rely either on rather ad-hoc, domain-specific semantic taxonomies, or on statistical models on large collections of unlabeled data. Recent results have shown that symbolic noun compound interpretation systems using machine learning techniques coupled with a large lexical hierarchy perform with very good accuracy, but they are most of the time tailored to a specific domain (Rosario and Hearst, 2001). On the other hand, the majority of corpus statistics approaches to noun compound interpretation collect statistics on the occurrence frequency of the noun constituents and use them in a probabilistic model (Lauer, 1995). More recently, (Lapata and Keller, 2004) showed that simple unsupervised models perform significantly better when the frequencies are obtained from the web, rather than from a large standard corpus. Other researchers (Pantel and Pennacchiotti, 2006), (Snow et al., 2006) use clustering techniques coupled with syntactic dependency features to identify IS-A relations in large text collections. (Kim and Baldwin, 2006) and (Turney, 2006) focus on the lexical similarity of unseen noun compounds with those found in training.

However, although the web-based solution might overcome the data sparseness problem, the current probabilistic models are limited by the lack of deep linguistic information. In this paper we investigate the role of cross-linguistic information in the task of English NP semantic interpretation and show the importance of a set of novel linguistic features.

3 Corpus Analysis

For a better understanding of the meaning of the N N and N P N instances, we analyzed the semantic behavior of these constructions on a large cross-linguistic corpora of examples. We are interested in what syntactic constructions are used to translate the English instances to the target Romance languages and vice-versa, what semantic relations do these constructions encode, and what is the corpus distribution of the semantic relations.

3.1 Lists of semantic classification relations

Although the NP interpretation problem has been studied for a long time, researchers haven't agreed on the number and the level of abstraction of these semantic categories. They can vary from a few prepositions (Lauer, 1995) to hundreds or thousands specific semantic relations (Finin, 1980). The more abstract the categories, the more noun phrases are covered, but also the more room for variation as to which category a phrase should be assigned.

In this paper we experiment with two state of the art classification sets used in NP interpretation. The first is a core set of 22 semantic relations (22 SRs) identified by us from the computational linguistics literature. This list, presented in Table 1 along with examples is general enough to cover a large majority of text semantics while keeping the semantic relations to a manageable number. The second set is Lauer's list of 8 prepositions (8 PP) and can be applied only to noun compounds (*of, for, with, in, on, at, about*, and *from* – e.g., according to this classification, *love story* can be classified as *story* `about` *love*). We selected these sets as they are of different size and contain semantic classification categories at different levels of abstraction. Lauer's list is more abstract and, thus capable of encoding a large number of noun compound instances, while the 22-SR list contains finer grained semantic categories. We show below the coverage of these semantic lists on two different corpora and how well they solve the interpretation problem of noun phrases.

3.2 The data

The data was collected from two text collections with different distributions and of different genre,

POSSESSION (family estate); KINSHIP (sister of the boy); PROPERTY (lubricant viscosity); AGENT (return of the natives); THEME (acquisition of stock); TEMPORAL (morning news); DEPICTION-DEPICTED (a picture of my niece); PART-WHOLE (brush hut); HYPERNYMY (IS-A) (daisy flower); CAUSE (scream of pain); MAKE/PRODUCE (chocolate factory); INSTRUMENT (laser treatment); LOCATION (castle in the desert); PURPOSE (cough syrup); SOURCE (grapefruit oil); TOPIC (weather report); MANNER (performance with passion); beneficiary (rights of citizens); MEANS (bus service); EXPERIENCER (fear of the girl); MEASURE (cup of sugar); TYPE (framework law);

Table 1: The list of 22 semantic relations (22-SRs).

Europarl[1] and CLUVI[2]. The Europarl data was assembled by combining the Spanish-English, Italian-English, French-English and Portuguese-English corpora which were automatically aligned based on exact matches of English translations. Then, we considered only the English sentences which appeared verbatim in all four language pairs. The resulting English corpus contained 10,000 sentences which were syntactically parsed (Charniak, 2000). From these we extracted the first 3,000 NP instances (N N: 48.82% and N P N: 51.18%).

CLUVI is an open text repository of parallel corpora of contemporary oral and written texts in some of the Romance languages. Here, we focused only on the English-Portuguese and English-Spanish parallel texts from the works of John Steinbeck, H. G. Wells, J. Salinger, and others. Using the CLUVI search interface we created a sentence-aligned parallel corpus of 2,800 English-Spanish and English-Portuguese sentences. The English versions were automatically parsed after which each N N and N P N instance thus identified was manually mapped to the corresponding translations. The resulting corpus contains 2,200 English instances with a distribution of 26.77% N N and 73.23% N P N.

3.3 Corpus Annotation

For each corpus, each NP instance was presented separately to two experienced annotators in a web interface in context along with the English sentence and its translations. Since the corpora do not cover some of the languages (Romanian in Europarl and CLUVI, and Italian and French in CLUVI), three other native speakers of these languages and fluent in English provided the translations which were added to the list. The two computational semantics annotators had to tag each English constituent noun with its corresponding WordNet sense and each instance with the corresponding semantic category. If the word was not found in WordNet the instance was not considered. Whenever the annotators found an example encoding a semantic category other than those provided or they didn't know what interpretation to give, they had to tag it as "OTHER-SR", and respectively "OTHER-PP"[3]. The details of the annotation task and the observations drawn from there are presented in a companion paper (Girju, 2007).

The corpus instances used in the corpus analysis phase have the following format: $<$NP$_{En}$;NP$_{Es}$; NP$_{It}$; NP$_{Fr}$; NP$_{Port}$; NP$_{Ro}$; target$>$. The word *target* is one of the 23 (22 + OTHER-SR) semantic relations and one of the eight prepositions considered or OTHER-PP (with the exception of those N P N instances that already contain a preposition). For example, $<$*development cooperation; cooperación para el desarrollo; cooperazione allo sviluppo; coopération au développement; cooperare pentru dezvoltare;* PURPOSE / FOR$>$.

The annotators' agreement was measured using Kappa statistics: $K = \frac{Pr(A)-Pr(E)}{1-Pr(E)}$, where $Pr(A)$ is the proportion of times the annotators agree and $Pr(E)$ is the probability of agreement by chance. The Kappa values were obtained on Europarl (N N: 0.80 for 8-PP and 0.61 for 22-SR; N P N: 0.67 for 22-SR) and CLUVI (N N: 0.77 for 8-PP and 0.56 for 22-SR; N P N: 0.68 for 22-SR). We also computed the number of pairs that were tagged with OTHER by both annotators for each semantic relation and preposition paraphrase, over the number of examples classified in that category by at least one of the judges (in Europarl: 91% for 8-PP and 78% for 22-SR; in CLUVI: 86% for 8-PP and 69% for 22-SR).

The agreement obtained on the Europarl corpus is

[1]http://www.isi.edu/koehn/europarl/. This corpus contains over 20 million words in eleven official languages of the European Union covering the proceedings of the European Parliament from 1996 to 2001.

[2]CLUVI - Linguistic Corpus of the University of Vigo - Parallel Corpus 2.1 - http://sli.uvigo.es/CLUVI/

[3]The annotated corpora resulted in this research is available at http://apfel.ai.uiuc.edu.

higher than the one on CLUVI on both classification sets. This is partially explained by the distribution of semantic relations in both corpora, as will be shown in the next subsection.

3.4 Cross-linguistic distribution of Syntactic Constructions

From the sets of 2,954 (Europarl) and 2,168 (CLUVI) instances resulted after annotation, the data show that over 83% of the translation patterns for both text corpora on all languages were of the type N N and N P N. However, while their distribution is balanced in the Europarl corpus (about 45%, with a 64% N P N – 26% N N ratio for Romanian), in CLUVI the N P N constructions occur in more than 85% of the cases (again, with the exception of Romanian – 50%). It is interesting to note here that some of the English NPs are translated into both noun–noun and noun–adjective compounds in the target languages. For example, *love affair* translates in Italian as *storia d'amore* or the noun–adjective compound *relazione amorosa*. There are also instances that have just one word correspondent in the target language (e.g., *ankle boot* is *bottine* in French). The rest of the data is encoded by other syntactic paraphrases (e.g., *bomb site* is *luogo dove è esplosa la bomba* (It.)). [4].

From the initial corpus we considered those English instances that had all the translations encoded only by N N and N P N. Out of these, we selected only 1,023 Europarl and 1,008 CLUVI instances encoded by N N and N P N in all languages considered and resulted after agreement.

4 Model

4.1 Feature space

We have identified and experimented with 13 NP features presented below. With the exceptions of features F1-F5 (Girju et al., 2005), all the other features are novel.

A. English Features

F1 and **F2.** *Noun semantic class* specifies the WordNet sense of the head (F1) and modifier noun (F2) and implicitly points to all its hypernyms. For example, the hypernyms of *car#1* are: {*motor vehicle*}, .. {*entity*}. This feature helps generalize over the semantic classes of the two nouns in the corpus.

F3 and **F4.** *WordNet derivationally related form* specifies if the head (F3) and the modifier (F4) nouns are related to a corresponding WordNet verb (e.g. *statement* derived from *to state*; *cry* from *to cry*).

F5. *Prepositional cues* that link the two nouns in an NP. These can be either simple or complex prepositions such as "*of*" or "*according to*". In case of N N instances, this feature is "–" (e.g., *framework law*).

F6 and F7. *Type of nominalized noun* indicates the specific class of nouns the head (F6) or modifier (F7) belongs to depending on the verb it derives from. First, we check if the noun is a nominalization. For English we used NomLex-Plus (Meyers et al., 2004) to map nouns to corresponding verbs.[5] For example, "*destruction of the city*", where *destruction* is a nominalization. F6 and F7 may overlap with features F3 and F4 which are used in case the noun to be checked does not have an entry in the NomLex-Plus dictionary. These features are of particular importance since they impose some constraints on the possible set of relations the instance can encode. They take the following values (identified based on list of verbs extracted from VerbNet (Kipper et al., 2000)):

a. Active form nouns which have an intrinsic active voice predicate-argument structure. (Giorgi and Longobardi, 1991) argue that in English this is a necessary restriction. Most of the time, they represent states of emotion, such as fear, desire, etc. These nouns mark their internal argument through *of* and require most of the time prepositions like *por* and not *de* when translated in Romance. Our observations on the Romanian translations (captured by features F12 and F13 below) show that the possible cases of ambiguity are solved by the type of syntactic construction used. For example, N N genitive-marked constructions are used for EXPERIENCER–encoding instances, while *N de N* or *N pentru N* (N for N) are used for other relations. Such examples are *the love of children* – THEME (and not *the love by the children*). (Giorgi and Longobardi, 1991) mention that with such nouns that resist passivisation,

[4] "*the place where the bomb is exploded*" (It.)

[5] NomLex-Plus is a hand-coded database of 5,000 verb nominalizations, de-adjectival, and de-adverbial nouns including the corresponding subcategorization frames (verb-argument structure information).

the preposition introducing the internal argument, even if it is *of*, has always a semantic content, and is not a bare case-marker realizing the genitive case.

b. Unaccusative (ergative) nouns which are derived from ergative verbs that take only internal arguments (e.g., not agentive ones). For example, the transitive verb *to disband* allows the subject to be deleted as in the following sentences (1) "*The lead singer disbanded the group in 1991.*" and (2) "*The group disbanded.*". Thus, the corresponding ergative nominalization *the disbandment of the group* encodes a THEME relation and not AGENT.

c. Unergative (intransitive) nouns are derived from intransitive verbs and take only AGENT semantic relations. For example, *the departure of the girl*.

d. Inherently passive nouns such as *the capture of the soldier*. These nouns, like the verbs they are derived from, assume a default AGENT (subject) and being transitive, associate to their internal argument (introduced by "of" in the example above) the THEME relation.

B. Romance Features

F8, F9, F10, F11 and **F12.** *Prepositional cues* that link the two nouns are extracted from each translation of the English instance: F8 (Es.), F9 (Fr.), F10 (It.), F11 (Port.), and F12 (Ro.). These can be either simple or complex prepositions (e.g., *de*, *in materia de* (Es.)) in all five Romance languages, or the Romanian genitival article *a/ai/ale*. In Romanian the genitive case is assigned by the definite article of the first noun to the second noun, case realized as a suffix if the second noun is preceded by the definite article or as one of the genitival articles *a/ai/ale*. For example, the noun phrase *the beauty of the girl* is translated as *frumuseţea fetei* (*beauty*-the *girl*-gen), and *the beauty of a girl* as *frumuseţea unei fete* (*beauty*-the gen *girl*). For N N instances, this feature is "–".

F13. *Noun inflection* is defined only for Romanian and shows if the modifier noun is inflected (indicates the genitive case). This feature is used to help differentiate between instances encoding IS-A and other semantic relations in N N compounds in Romanian. It also helps in features F6 and F7, case a) when the choice of syntactic construction reflects different semantic content. For example, *iubirea pentru copii* (N P N) (*the love for children*) and not *iubirea copiilor* (N N) (*love expressed by the children*).

4.2 Learning Models

We have experimented with the support vector machines (SVM) model[6] and compared the results against two state-of-the-art models: a supervised model, Semantic Scattering (SS), (Moldovan and Badulescu, 2005), and a web-based unsupervised model (Lapata and Keller, 2004). The SVM and SS models were trained and tested on the Europarl and CLUVI corpora using a 8:2 ratio. The test dataset was randomly selected from each corpus and the test nouns (only for English) were tagged with the corresponding sense in context using a state of the art WSD tool (Mihalcea and Faruque, 2004).

After the initial NP instances in the training and test corpora were expanded with the corresponding features, we had to prepare them for SVM and SS. The method consists of a set of automatic iterative procedures of specialization of the English nouns on the WordNet IS-A hierarchy. Thus, after a set of necessary specialization iterations, the method produces specialized examples which through supervised machine learning are transformed into sets of semantic rules. This specialization procedure improves the system's performance since it efficiently separates the positive and negative noun-noun pairs in the WordNet hierarchy.

Initially, the training corpus consists of examples in the format exemplified by the feature space. Note that for the English NP instances, each noun constituent was expanded with the corresponding WordNet top semantic class. At this point, the generalized training corpus contains two types of examples: unambiguous and ambiguous. The second situation occurs when the training corpus classifies the same noun – noun pair into more than one semantic category. For example, both relationships "*chocolate cake*"-PART-WHOLE and "*chocolate article*"-TOPIC are mapped into the more general type <*entity#1, entity#1,* PART-WHOLE/TOPIC>[7]. We recursively specialize these examples to eliminate the ambiguity. By specialization, the semantic class is replaced with the corresponding hyponym for that particular sense, i.e. the concept immediately below in the hierarchy. These steps are repeated until there are no

[6] We used the package LIBSVM with a radial-based kernel *http://www.csie.ntu.edu.tw/∼cjlin/libsvm/*

[7] The specialization procedure applies only to features 1, 2.

more ambiguous examples. For the example above, the specialization stops at the first hyponym of *entity*: *physical entity* (for *cake*) and *abstract entity* (for *article*). For the unambiguous examples in the generalized training corpus (those that are classified with a single semantic relation), constraints are determined using cross validation on SVM.

A. Semantic Scattering uses a training data set to establish a boundary G^* on WordNet noun hierarchies such that each feature pair of noun – noun senses f_{ij} on this boundary maps uniquely into one of a predefined list of semantic relations, and any feature pair above the boundary maps into more than one semantic relation. For any new pair of noun–noun senses, the model finds the closest WordNet boundary pair.

The authors define with $SC^m = \{f_i^m\}$ and $SC^h = \{f_j^h\}$ the sets of semantic class features for modifier noun and, respectively head noun. A pair of <modifier – head> nouns maps uniquely into a semantic class feature pair $< f_i^m, f_j^h >$, denoted as f_{ij}. The probability of a semantic relation *r* given feature pair f_{ij}, $P(r|f_{ij}) = \frac{n(r,f_{ij})}{n(f_{ij})}$, is defined as the ratio between the number of occurrences of a relation *r* in the presence of feature pair f_{ij} over the number of occurrences of feature pair f_{ij} in the corpus. The most probable semantic relation $\hat{r}$ is $\arg\max_{r \in R} P(r|f_{ij}) = \arg\max_{r \in R} P(f_{ij}|r)P(r)$.

B. (Lapata and Keller, 2004)'s web-based unsupervised model classifies noun - noun instances based on Lauer's list of 8 prepositions and uses the web as training corpus. They show that the best performance is obtained with the trigram model $f(n_1, p, n_2)$. The count used for a given trigram is the number of pages returned by Altavista on the trigram corresponding queries. For example, for the test instance *war stories*, the best number of hits was obtained with the query *stories about war*.

For the Europarl and CLUVI test sets, we replicated Lapata & Keller's experiments using Google[8]. We formed inflected queries with the patterns they proposed and searched the web.

[8]As Google limits the number of queries to 1,000 per day, we repeated the experiment for a number of days. Although (Lapata and Keller, 2004) used Altavista in their experiments, they showed there is almost no difference between the correlations achieved using Google and Altavista counts.

5 Experimental results

Table 2 shows the results obtained against SS and Lapata & Keller's model on both corpora and the contribution the features exemplified in one baseline and six versions of the SVM model. The baseline is defined only for the English part of the NP feature set and measures the the contribution of the WordNet IS-A lexical hierarchy specialization. The baseline does not differentiate between unambiguous and ambiguous training examples (after just one level specialization) and thus, does not specialize the ambiguous ones. Moreover, here we wanted to see what is the difference between SS and SVM, and what is the contribution of the other English features, such as preposition and nominalization (F1–F7).

The table shows that, overall the performance is better for the Europarl corpus than for CLUVI. For the Baseline and SVM_1, SS [F1 + F2] gives better results than SVM. The inclusion of other English features (SVM [F1–F7]) adds more than 15% (with a higher increase in Europarl) for SVM_1.

The contribution of Romance linguistic features. Since our intuition is that the more translations are provided for an English noun phrase instance, the better the results, we wanted to see what is the impact of each Romance language on the overall performance. Thus, SVM_2 shows the results obtained for English and the Romance language that contributed the least to the performance (F1–F12). Here we computed the performance on all five English – Romance language combinations and chose the Romance language that provided the best result. Thus, SVM #2, #3, #4, #5, and #6 add Spanish, French, Italian, Portuguese, and Romanian in this order and show the contribution of each Romance preposition and all features for English.

The language ranking in Table 2 shows that Romance languages considered here have a different contribution to the overall performance. While the addition of Italian in Europarl decreases the performance, Portuguese doesn't add anything. However, a closer analysis of the data shows that this is mostly due to the distribution of the corpus instances. For example, French, Italian, Spanish, and Portuguese are most of the time consistent in the choice of preposition (e.g. most of the time, if the preposition 'de' ('of') is used in French, then the

Learning models		Results [%]			
		CLUVI		Europarl	
		8-PP	22-SR	8-PP	22-SR
Baseline (En.) (no specializ.)	SS (F1+F2)	44.11	48.03	38.7	38
	SVM (F1+F2)	36.37	40.67	31.18	34.81
	SVM (F1-F7)	–	52.15	–	47.37
SVM$_1$ (En.)	SS (F1+F2)	56.22	61.33	53.1	56.81
	SVM (F1+F2)	45.08	46.1	40.23	42.2
	SVM (F1-F7)	–	62.54	–	74.19
SVM$_2$ (En. + Es.)	SVM (F1-F8)	–	64.18	–	75.74
SVM$_3$ (En.+Es.+Fr.)	SVM (F1-F9)	–	67.8	–	76.52
SVM$_4$ (En.+Es.+Fr.+It.)	SVM (F1-F10)	–	66.31	–	75.74
SVM$_5$ (En.+Es.+Fr.+It+Port.)	SVM (F1-F11)	–	67.12	–	75.74
SVM$_6$ (En.+Romance: F1–F13)		–	**74.31**	–	**77.9**
Lapata & Keller's unsupervised model (En.)		44.15	–	45.31	–

Table 2: The performance of the cross-linguistic SVM models compared against one baseline, SS model and Lapata & Keller's unsupervised model. *Accuracy* (number of correctly labeled instances over the number of instances in the test set).

corresponding preposition is used in the other four language translations). A notable exception here is Romanian which provides two possible constructions: the N P N and the genitive-marked N N. The table shows (in the increase in performance between SVM_5 and SVM_6) that this choice is not random, but influenced by the meaning of the instances (features F12, F13). This observation is also supported by the contribution of each feature to the overall performance. For example, in Europarl, the WordNet verb and nominalization features of the head noun (F3, F6) have a contribution of 4.08%, while for the modifier nouns it decreases by about 2%. The preposition (F5) contributes 4.41% (Europarl) and 5.24% (CLUVI) to the overall performance.

A closer analysis of the data shows that in Europarl most of the N N instances were naming noun compounds such as *framework law* (TYPE) and, most of the time, are encoded by N N patterns in the target languages (e.g., *legge quadro* (It.)). In the CLUVI corpus, on the other hand, the N N Romance translations represented only 1% of the data. A notable exception here is Romanian where most NPs are represented as genitive–marked noun compounds. However, there are instances that are encoded mostly or only as N P N constructions and this choice correlates with the meaning of the instance. For example, *the milk glass* (PURPOSE) translates as *paharul de lapte* (*glass*-the *of milk*) and not as *paharul laptelui* (*glass*-the *milk*-gen), *the olive oil* (SOURCE) translates as *uleiul de măsline* (*oil*-the *of olive*) and not as *uleiul măslinei* (*oil*-the *olive*-gen). Other examples include CAUSE and TOPIC.

Lauer's set of 8 prepositions represents 94.5% (Europarl) and 97% (CLUVI) of the N P N instances. From these, the most frequent preposition is "of" with a coverage of 70.31% (Europarl) and 85.08% (CLUVI). Moreover, in the Europarl corpus, 26.39% of the instances are synthetic phrases (where one of the nouns is a nominalization) encoding AGENT, EXPERIENCER, THEME, BENEFICIARY. Out of these instances, 74.81% use the preposition *of*. In CLUVI, 11.71% of the examples were verbal, from which the preposition *of* has a coverage of 82.20%. The many-to-many mappings of the prepositions (especially *of/de*) to the semantic classes adds to the complexity of the interpretation task. Thus, for the interpretation of these constructions a system must rely on the semantic information of the preposition and two constituent nouns in particular, and on context in general.

In Europarl, the most frequently occurring relations are PURPOSE, TYPE, and THEME that together represent about 57% of the data followed by PART-WHOLE, PROPERTY, TOPIC, AGENT, and LOCATION with an average coverage of about 6.23%. Moreover, other relations such as KINSHIP, DEPICTION, MANNER, MEANS did not occur in this corpus and 5.08% represented OTHER-SR relations. This semantic distribution contrasts with the one in CLUVI, which uses a more descriptive language. Here, the most frequent relation by far

is PART-WHOLE (32.14%), followed by LOCATION (12.40%), THEME (9.23%) and OTHER-SR (7.74%). It is interesting to note here that only 5.70% of the TYPE relation instances in Europarl were unique. This is in contrast with the other relations in both corpora, where instances were mostly unique.

We also report here our observations on Lapata & Keller's unsupervised model. An analysis of these results showed that the order of the constituent nouns in the N P N paraphrase plays an important role. For example, a search for *blood vessels* generated similar frequency counts for *vessels of blood* and *blood in vessels*. About 30% noun - noun paraphrasable pairs preserved the order in the corresponding N P N paraphrases. We also manually checked the first five entries generated by Google for each most frequent prepositional paraphrase for 50 instances and noticed that about 35% of them were wrong due to syntactic and/or semantic ambiguities. Thus, since we wanted to measure the impact of these ambiguities of noun compounds on the interpretation performance, we further tested the probabilistic web-based model on four distinct test sets selected from Europarl, each containing 30 noun - noun pairs encoding different types of ambiguity: in set#1 the noun constituents had only one part of speech and one WordNet sense; in set#2 the nouns had at least two possible parts of speech and were semantically unambiguous, in set#3 the nouns were ambiguous only semantically, and in set#4 they were ambiguous both syntactically and semantically. For unambiguous noun-noun pairs (set#1), the model obtained an accuracy of 35.01%, while for more semantically ambiguous compounds it obtained an accuracy of about 48.8%. This shows that for more semantically ambiguous noun - noun pairs, the web-based probabilistic model introduces a significant number of false positives. Thus, the more abstract the categories, the more noun compounds are covered, but also the more room for variation as to which category a compound should be assigned.

6 Discussion and Conclusions

In this paper we presented a supervised, knowledge-intensive interpretation model which takes advantage of new linguistic information from English and a list of five Romance languages. Our approach to NP interpretation is novel in several ways. We defined the problem in a cross-linguistic framework and provided empirical observations on the distribution of the syntax and meaning of noun phrases on two different corpora based on two state-of-the-art classification tag sets.

As future work we consider the inclusion of other features such as the semantic classes of Romance nouns from aligned EuroWordNets, and other sentence features. Since the results obtained can be seen as an upper bound on NP interpretation due to perfect English - Romance NP alignment, we will experiment with automatic translations generated for the test data. Moreover, we like to extend the analysis to other set of languages whose structures are very different from English and Romance.

References

T. W. Finin. 1980. *The Semantic Interpretation of Compound Nominals*. Ph.D. thesis, University of Illinois at Urbana-Champaign.

A. Giorgi and G. Longobardi. 1991. The syntax of noun phrases. Cambridge University Press.

R. Girju, D. Moldovan, M. Tatu, and D. Antohe. 2005. On the semantics of noun compounds. *Computer Speech and Language*, 19(4):479–496.

R. Girju. 2007. Experiments with an annotation scheme for a knowledge-rich noun phrase interpretation system. The Linguistic Annotation Workshop at ACL, Prague.

Su Nam Kim and T. Baldwin. 2006. Interpreting semantic relations in noun compounds via verb semantics. *COLING-ACL*.

K. Kipper, H. Dong, and M. Palmer. 2000. Class-based construction of a verb lexicon. *AAAI* Conference, Austin.

M. Lapata and F. Keller. 2004. The Web as a baseline: Evaluating the performance of unsupervised Web-based models for a range of NLP tasks. *HLT-NAACL*.

M. Lauer. 1995. Corpus statistics meet the noun compound: Some empirical results. *ACL*, Cambridge, Mass.

A. Meyers, R. Reeves, C. Macleod, R. Szekeley V. Zielinska, and B. Young. 2004. The cross-breeding of dictionaries. *LREC-2004*, Lisbon, Portugal.

R. Mihalcea and E. Faruque. 2004. Senselearner: Minimally supervised word sense disambiguation for all words in open text. *ACL/SIGLEX Senseval-3*, Barcelona, Spain.

D. Moldovan and A. Badulescu. 2005. A semantic scattering model for the automatic interpretation of genitives. *HLT/EMNLP Conference*, Vancouver, Canada.

P. Pantel and M. Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. *COLING/ACL*, Sydney, Australia.

B. Rosario and M. Hearst. 2001. Classifying the semantic relations in noun compounds. *EMNLP Conference*.

R. Snow, D. Jurafsky, and A. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. *COLING-ACL*.

P. Turney. 2006. Expressing implicit semantic relations without supervision. *COLING/ACL*, Sydney, Australia.

Learning to Extract Relations from the Web using Minimal Supervision

Razvan C. Bunescu
Department of Computer Sciences
University of Texas at Austin
1 University Station C0500
Austin, TX 78712
razvan@cs.utexas.edu

Raymond J. Mooney
Department of Computer Sciences
University of Texas at Austin
1 University Station C0500
Austin, TX 78712
mooney@cs.utexas.edu

Abstract

We present a new approach to relation extraction that requires only a handful of training examples. Given a few pairs of named entities known to exhibit or not exhibit a particular relation, bags of sentences containing the pairs are extracted from the web. We extend an existing relation extraction method to handle this weaker form of supervision, and present experimental results demonstrating that our approach can reliably extract relations from web documents.

1 Introduction

A growing body of recent work in information extraction has addressed the problem of *relation extraction* (RE), identifying relationships between entities stated in text, such as `LivesIn(Person, Location)` or `EmployedBy(Person, Company)`. Supervised learning has been shown to be effective for RE (Zelenko et al., 2003; Culotta and Sorensen, 2004; Bunescu and Mooney, 2006); however, annotating large corpora with examples of the relations to be extracted is expensive and tedious.

In this paper, we introduce a supervised learning approach to RE that requires only a handful of training examples and uses the web as a corpus. Given a few pairs of well-known entities that clearly exhibit or do not exhibit a particular relation, such as `CorpAcquired`(Google, YouTube) and `not`(`CorpAcquired`(Yahoo, Microsoft)), a search engine is used to find sentences on the web that mention both of the entities in each of the pairs. Although not all of the sentences for positive pairs will state the desired relationship, many of them will. Presumably, none of the sentences for negative pairs state the targeted relation. *Multiple instance learning* (MIL) is a machine learning framework that exploits this sort of weak supervision, in which a *positive bag* is a set of instances which is guaranteed to contain at least one positive example, and a *negative bag* is a set of instances all of which are negative. MIL was originally introduced to solve a problem in biochemistry (Dietterich et al., 1997); however, it has since been applied to problems in other areas such as classifying image regions in computer vision (Zhang et al., 2002), and text categorization (Andrews et al., 2003; Ray and Craven, 2005).

We have extended an existing approach to relation extraction using support vector machines and string kernels (Bunescu and Mooney, 2006) to handle this weaker form of MIL supervision. This approach can sometimes be misled by textual features correlated with the specific entities in the few training pairs provided. Therefore, we also describe a method for weighting features in order to focus on those correlated with the target relation rather than with the individual entities. We present experimental results demonstrating that our approach is able to accurately extract relations from the web by learning from such weak supervision.

2 Problem Description

We address the task of learning a relation extraction system targeted to a fixed binary relationship R. The only supervision given to the learning algo-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 576–583,
Prague, Czech Republic, June 2007.

rithm is a small set of pairs of named entities that are known to belong (positive) or not belong (negative) to the given relationship. Table 1 shows four positive and two negative example pairs for the corporate acquisition relationship. For each pair, a bag of sentences containing the two arguments can be extracted from a corpus of text documents. The corpus is assumed to be sufficiently large and diverse such that, if the pair is positive, it is highly likely that the corresponding bag contains at least one sentence that explicitly asserts the relationship R between the two arguments. In Section 6 we describe a method for extracting bags of relevant sentences from the web.

$+/-$	**Arg** a_1	**Arg** a_2
$+$	Google	YouTube
$+$	Adobe Systems	Macromedia
$+$	Viacom	DreamWorks
$+$	Novartis	Eon Labs
$-$	Yahoo	Microsoft
$-$	Pfizer	Teva

Table 1: Corporate Acquisition Pairs.

Using a limited set of entity pairs (e.g. those in Table 1) and their associated bags as training data, the aim is to induce a relation extraction system that can reliably decide whether two entities mentioned in the same sentence exhibit the target relationship or not. In particular, when tested on the example sentences from Figure 1, the system should classify S_1, S_3,and S_4 as positive, and S_2 and S_5 as negative.

$+/S_1$: Search engine giant **Google** has bought video-sharing website **YouTube** in a controversial $1.6 billion deal.

$-/S_2$: The companies will merge **Google**'s search expertise with **YouTube**'s video expertise, pushing what executives believe is a hot emerging market of video offered over the Internet.

$+/S_3$: **Google** has acquired social media company, **YouTube** for $1.65 billion in a stock-for-stock transaction as announced by Google Inc. on October 9, 2006.

$+/S_4$: Drug giant **Pfizer Inc.** has reached an agreement to buy the private biotechnology firm **Rinat Neuroscience Corp.**, the companies announced Thursday.

$-/S_5$: He has also received consulting fees from Alpharma, Eli Lilly and Company, **Pfizer**, Wyeth Pharmaceuticals, **Rinat Neuroscience**, Elan Pharmaceuticals, and Forest Laboratories.

Figure 1: Sentence examples.

As formulated above, the learning task can be seen as an instance of *multiple instance learning*. However, there are important properties that set it apart from problems previously considered in MIL. The most distinguishing characteristic is that the number of bags is very small, while the average size of the bags is very large.

3 Multiple Instance Learning

Since its introduction by Dietterich (1997), an extensive and quite diverse set of methods have been proposed for solving the MIL problem. For the task of relation extraction, we consider only MIL methods where the decision function can be expressed in terms of kernels computed between bag instances. This choice was motivated by the comparatively high accuracy obtained by kernel-based SVMs when applied to various natural language tasks, and in particular to relation extraction. Through the use of kernels, SVMs (Vapnik, 1998; Schölkopf and Smola, 2002) can work efficiently with instances that implicitly belong to a high dimensional feature space. When used for classification, the decision function computed by the learning algorithm is equivalent to a hyperplane in this feature space. Overfitting is avoided in the SVM formulation by requiring that positive and negative training instances be maximally separated by the decision hyperplane.

Gartner *et al.* (2002) adapted SVMs to the MIL setting using various multi-instance kernels. Two of these – the normalized set kernel, and the statistic kernel – have been experimentally compared to other methods by Ray and Craven (2005), with competitive results. Alternatively, a simple approach to MIL is to transform it into a standard supervised learning problem by labeling all instances from positive bags as positive. An interesting outcome of the study conducted by Ray and Craven (2005) was that, despite the class noise in the resulting positive examples, such a simple approach often obtains competitive results when compared against other more sophisticated MIL methods.

We believe that an MIL method based on multi-instance kernels is not appropriate for training datasets that contain just a few, very large bags. In a multi-instance kernel approach, only bags (and not instances) are considered as training examples,

which means that the number of support vectors is going to be upper bounded by the number of training bags. Taking the bags from Table 1 as a sample training set, the decision function is going to be specified by at most seven parameters: the coefficients for at most six support vectors, plus an optional bias parameter. A hypothesis space characterized by such a small number of parameters is likely to have insufficient capacity.

Based on these observations, we decided to transform the MIL problem into a standard supervised problem as described above. The use of this approach is further motivated by its simplicity and its observed competitive performance on very diverse datasets (Ray and Craven, 2005). Let $\mathcal{X}$ be the set of bags used for training, $\mathcal{X}_p \subseteq \mathcal{X}$ the set of positive bags, and $\mathcal{X}_n \subseteq \mathcal{X}$ the set of negative bags. For any instance $x \in X$ from a bag $X \in \mathcal{X}$, let $\phi(x)$ be the (implicit) feature vector representation of x. Then the corresponding SVM optimization problem can be formulated as in Figure 2:

minimize:

$$\mathbf{J}(w, b, \xi) = \tfrac{1}{2}\|w\|^2 + \frac{C}{L}\left(c_p \tfrac{L_n}{L} \Xi_p + c_n \tfrac{L_p}{L} \Xi_n\right)$$

$$\Xi_p = \sum_{X \in \mathcal{X}_p} \sum_{x \in X} \xi_x$$

$$\Xi_n = \sum_{X \in \mathcal{X}_n} \sum_{x \in X} \xi_x$$

subject to:

$$w\,\phi(x) + b \geq +1 - \xi_x, \;\; \forall x \in X \in \mathcal{X}_p$$

$$w\,\phi(x) + b \leq -1 + \xi_x, \;\; \forall x \in X \in \mathcal{X}_n$$

$$\xi_x \geq 0$$

Figure 2: SVM Optimization Problem.

The capacity control parameter C is normalized by the total number of instances $L = L_p + L_n = \sum_{X \in \mathcal{X}_p} |X| + \sum_{X \in \mathcal{X}_n} |X|$, so that it remains independent of the size of the dataset. The additional non-negative parameter c_p ($c_n = 1 - c_p$) controls the relative influence that false negative vs. false positive errors have on the value of the objective function. Because not all instances from positive bags are real positive instances, it makes sense to have false negative errors be penalized less than false positive errors (i.e. $c_p < 0.5$).

In the dual formulation of the optimization problem from Figure 2, bag instances appear only inside dot products of the form $K(x_1, x_2) = \phi(x_1)\phi(x_2)$. The kernel K is instantiated to a subsequence kernel, as described in the next section.

4 Relation Extraction Kernel

The training bags consist of sentences extracted from online documents, using the methodology described in Section 6. Parsing web documents in order to obtain a syntactic analysis often gives unreliable results – the type of narrative can vary greatly from one web document to another, and sentences with grammatical errors are frequent. Therefore, for the initial experiments, we used a modified version of the subsequence kernel of Bunescu and Mooney (2006), which does not require syntactic information. This kernel computes the number of common subsequences of tokens between two sentences. The subsequences are constrained to be "anchored" at the two entity names, and there is a maximum number of tokens that can appear in a sequence. For example, a subsequence feature for the sentence S_1 in Figure 1 is $\tilde{s}$ = "$\langle e_1 \rangle$ ... *bought* ... $\langle e_2 \rangle$ ... *in* ... *billion* ... *deal*", where $\langle e_1 \rangle$ and $\langle e_2 \rangle$ are generic placeholders for the two entity names. The subsequence kernel induces a feature space where each dimension corresponds to a sequence of words. Any such sequence that matches a subsequence of words in a sentence example is down-weighted as a function of the total length of the gaps between every two consecutive words. More exactly, let $s = w_1 w_2 \ldots w_k$ be a sequence of k words, and $\tilde{s} = w_1\, g_1\, w_2\, g_2 \ldots w_{k-1}\, g_{k-1}\, w_k$ a matching subsequence in a relation example, where g_i stands for any sequence of words between w_i and w_{i+1}. Then the sequence s will be represented in the relation example as a feature with weight computed as $\tau(s) = \lambda^{g(\tilde{s})}$. The parameter λ controls the magnitude of the gap penalty, where $g(\tilde{s}) = \sum_i |g_i|$ is the total gap.

Many relations, like the ones that we explore in the experimental evaluation, cannot be expressed without using at least one content word. We therefore modified the kernel computation to optionally ignore subsequence patterns formed exclusively of

stop words and punctuation signs. In Section 5.1, we introduce a new weighting scheme, wherein a weight is assigned to every token. Correspondingly, every sequence feature will have an additional multiplicative weight, computed as the product of the weights of all the tokens in the sequence. The aim of this new weighting scheme, as detailed in the next section, is to eliminate the bias caused by the special structure of the relation extraction MIL problem.

5 Two Types of Bias

As already hinted at the end of Section 2, there is one important property that distinguishes the current MIL setting for relation extraction from other MIL problems: the training dataset contains very few bags, and each bag can be very large. Consequently, an application of the learning model described in Sections 3 & 4 is bound to be affected by the following two types of bias:

■ **[Type I Bias]** By definition, all sentences inside a bag are constrained to contain the same two arguments. Words that are semantically correlated with either of the two arguments are likely to occur in many sentences. For example, consider the sentences S_1 and S_2 from the bag associated with "Google" and "YouTube" (as shown in Figure 1). They both contain the words "search" – highly correlated with "Google", and "video" – highly correlated with "YouTube", and it is likely that a significant percentage of sentences in this bag contain one of the two words (or both). The two entities can be mentioned in the same sentence for reasons other than the target relation R, and these noisy training sentences are likely to contain words that are correlated with the two entities, without any relationship to R. A learning model where the features are based on words, or word sequences, is going to give too much weight to words or combinations of words that are correlated with either of individual arguments. This overweighting will adversely affect extraction performance through an increased number of errors. A method for eliminating this type of bias is introduced in Section 5.1.

■ **[Type II Bias]** While Type I bias is due to words that are correlated with the arguments of a relation instance, the Type II bias is caused by words that are specific to the relation instance itself. Using FrameNet terminology (Baker et al., 1998), these correspond to instantiated frame elements. For example, the corporate acquisition frame can be seen as a subtype of the "Getting" frame in FrameNet. The *core* elements in this frame are the *Recipient* (e.g. Google) and the *Theme* (e.g. YouTube), which for the acquisition relationship coincide with the two arguments. They do not contribute any bias, since they are replaced with the generic tags $\langle e_1 \rangle$ and $\langle e_2 \rangle$ in all sentences from the bag. There are however other frame elements – *peripheral*, or *extra-thematic* – that can be instantiated with the same value in many sentences. In Figure 1, for instance, sentence S_3 contains two non-core frame elements: the *Means* element (e.g "in a stock-for-stock transaction") and the *Time* element (e.g. "on October 9, 2006"). Words from these elements, like "stock", or "October", are likely to occur very often in the Google-YouTube bag, and because the training dataset contains only a few other bags, subsequence patterns containing these words will be given too much weight in the learned model. This is problematic, since these words can appear in many other frames, and thus the learned model is likely to make errors. Instead, we would like the model to focus on words that trigger the target relationship (in FrameNet, these are the lexical units associated with the target frame).

5.1 A Solution for Type I Bias

In order to account for how strongly the words in a sequence are correlated with either of the individual arguments of the relation, we modify the formula for the sequence weight $\tau(s)$ by factoring in a weight $\tau(w)$ for each word in the sequence, as illustrated in Equation 1.

$$\tau(s) = \lambda^{g(\bar{s})} \cdot \prod_{w \in s} \tau(w) \qquad (1)$$

Given a predefined set of weights $\tau(w)$, it is straightforward to update the recursive computation of the subsequence kernel so that it reflects the new weighting scheme.

If all the word weights are set to 1, then the new kernel is equivalent to the old one. What we want, however, is a set of weights where words that are correlated with either of the two arguments are given lower weights. For any word, the decrease in weight

should reflect the degree of correlation between that word and the two arguments. Before showing the formula used for computing the word weights, we first introduce some notation:

- Let $X \in \mathcal{X}$ be an arbitrary bag, and let $X.a_1$ and $X.a_2$ be the two arguments associated with the bag.
- Let $C(X)$ be the size of the bag (i.e. the number of sentences in the bag), and $C(X, w)$ the number of sentences in the bag X that contain the word w. Let $P(w|X) = C(X, w)/C(X)$.
- Let $P(w|X.a_1 \vee X.a_2)$ be the probability that the word w appears in a sentence due only to the presence of $X.a_1$ or $X.a_2$, assuming $X.a_1$ and $X.a_2$ are independent causes for w.

The word weights are computed as follows:

$$\begin{aligned}\tau(w) &= \frac{C(X,w) - P(w|X.a_1 \vee X.a_2) \cdot C(X)}{C(X,w)} \\ &= 1 - \frac{P(w|X.a_1 \vee X.a_2)}{P(w|X)} \qquad (2)\end{aligned}$$

The quantity $P(w|X.a_1 \vee X.a_2) \cdot C(X)$ represents the expected number of sentences in which w would occur, if the only causes were $X.a_1$ or $X.a_2$, independent of each other. We want to discard this quantity from the total number of occurrences $C(X, w)$, so that the effect of correlations with $X.a_1$ or $X.a_2$ is eliminated.

We still need to compute $P(w|X.a_1 \vee X.a_2)$. Because in the definition of $P(w|X.a_1 \vee X.a_2)$, the arguments $X.a_1$ and $X.a_2$ were considered independent causes, $P(w|X.a_1 \vee X.a_2)$ can be computed with the *noisy-or* operator (Pearl, 1986):

$$\begin{aligned}P(\cdot) &= 1-(1-P(w|a_1)) \cdot (1-P(w|a_2)) \qquad (3) \\ &= P(w|a_1)+P(w|a_2)-P(w|a_1) \cdot P(w|a_2)\end{aligned}$$

The quantity $P(w|a)$ represents the probability that the word w appears in a sentence due only to the presence of a, and it could be estimated using counts on a sufficiently large corpus. For our experimental evaluation, we used the following approximation: given an argument a, a set of sentences containing a are extracted from web documents (details in Section 6). Then $P(w|a)$ is simply approximated with the ratio of the number of sentences containing w over the total number of sentences, i.e. $P(w|a) = C(w, a)/C(a)$. Because this may be an overestimate (w may appear in a sentence containing a due to causes other than a), and also because of data sparsity, the quantity $\tau(w)$ may sometimes result in a negative value – in these cases it is set to 0, which is equivalent to ignoring the word w in all subsequence patterns.

6 MIL Relation Extraction Datasets

For the purpose of evaluation, we created two datasets: one for corporate acquisitions, as shown in Table 2, and one for the person-birthplace relation, with the example pairs from Table 3. In both tables, the top part shows the training pairs, while the bottom part shows the test pairs.

+/−	**Arg** a_1	**Arg** a_2	**Size**
+	Google	YouTube	1375
+	Adobe Systems	Macromedia	622
+	Viacom	DreamWorks	323
+	Novartis	Eon Labs	311
−	Yahoo	Microsoft	163
−	Pfizer	Teva	247
+	Pfizer	Rinat Neuroscience	50 (41)
+	Yahoo	Inktomi	433 (115)
−	Google	Apple	281
−	Viacom	NBC	231

Table 2: Corporate Acquisition Pairs.

+/−	**Arg** a_1	**Arg** a_2	**Size**
+	Franz Kafka	Prague	552
+	Andre Agassi	Las Vegas	386
+	Charlie Chaplin	London	292
+	George Gershwin	New York	260
−	Luc Besson	New York	74
−	Wolfgang A. Mozart	Vienna	288
+	Luc Besson	Paris	126 (6)
+	Marie Antoinette	Vienna	105 (39)
−	Charlie Chaplin	Hollywood	266
−	George Gershwin	London	104

Table 3: Person-Birthplace Pairs.

Given a pair of arguments (a_1, a_2), the corresponding bag of sentences is created as follows:

■ A query string "$a_1 * * * * * * * a_2$" containing seven wildcard symbols between the two arguments is submitted to Google. The preferences are set to search only for pages written in English, with Safesearch turned on. This type of query will match documents where an occurrence of a_1 is separated from an occurrence of a_2 by at most seven content words. This is an approximation of our actual information

need: "return all documents containing a_1 and a_2 in the same sentence".

■ The returned documents (limited by Google to the first 1000) are downloaded, and then the text is extracted using the HTML parser from the Java Swing package. Whenever possible, the appropriate HTML tags (e.g. *BR, DD, P*, etc.) are used as hard end-of-sentence indicators. The text is further segmented into sentences with the OpenNLP[1] package.

■ Sentences that do not contain both arguments a_1 and a_2 are discarded. For every remaining sentence, we find the occurrences of a_1 and a_2 that are closest to each other, and create a relation example by replacing a_1 with $\langle e_1 \rangle$ and a_2 with $\langle e_2 \rangle$. All other occurrences of a_1 and a_2 are replaced with a null token ignored by the subsequence kernel.

The number of sentences in every bag is shown in the last column of Tables 2 & 3. Because Google also counts pages that are deemed too similar in the first 1000, some of the bags can be relatively small.

As described in Section 5.1, the word-argument correlations are modeled through the quantity $P(w|a) = C(w,a)/C(a)$, estimated as the ratio between the number of sentences containing w and a, and the number of sentences containing a. These counts are computed over a bag of sentences containing a, which is created by querying Google for the argument a, and then by processing the results as described above.

7 Experimental Evaluation

Each dataset is split into two sets of bags: one for training and one for testing. The test dataset was purposefully made difficult by including negative bags with arguments that during training were used in positive bags, and vice-versa. In order to evaluate the relation extraction performance at the sentence level, we manually annotated all instances from the positive test bags. The last column in Tables 2 & 3 shows, between parentheses, how many instances from the positive test bags are real positive instances. The corporate acquisition test set has a total of 995 instances, out of which 156 are positive. The person-birthplace test set has a total of 601 instances, and only 45 of them are positive. Extrapolating from the test set distribution, the positive bags in the person-birthplace dataset are significantly sparser in real positive instances than the positive bags in the corporate acquisition dataset.

The subsequence kernel described in Section 4 was used as a custom kernel for the LibSVM[2] Java package. When run with the default parameters, the results were extremely poor – too much weight was given to the slack term in the objective function. Minimizing the regularization term is essential in order to capture subsequence patterns shared among positive bags. Therefore LibSVM was modified to solve the optimization problem from Figure 2, where the capacity parameter C is normalized by the size of the transformed dataset. In this new formulation, C is set to its default value of 1.0 – changing it to other values did not result in significant improvement. The trade-off between false positive and false negative errors is controlled by the parameter c_p. When set to its default value of 0.5, false-negative errors and false positive errors have the same impact on the objective function. As expected, setting c_p to a smaller value (0.1) resulted in better performance. Tests with even lower values did not improve the results.

We compare the following four systems:

■ **SSK–MIL**: This corresponds to the MIL formulation from Section 3, with the original subsequence kernel described in Section 4.

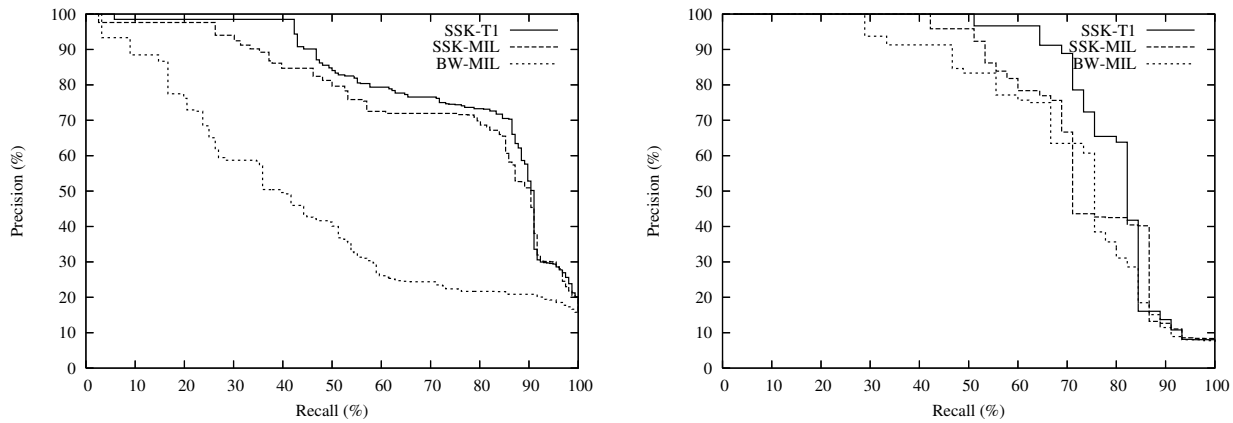
■ **SSK–T1**: This is the SSK–MIL system augmented with word weights, so that the Type I bias is reduced, as described in Section 5.1.

■ **BW-MIL**: This is a bag-of-words kernel, in which the relation examples are classified based on the unordered words contained in the sentence. This baseline shows the performance of a standard text-classification approach to the problem using a state-of-the art algorithm (SVM).

■ **SSK–SIL**: This corresponds to the original subsequence kernel trained with traditional, single instance learning (SIL) supervision. For evaluation, we train on the manually labeled instances from the test bags. We use a combination of one positive bag and one negative bag for training, while the other two bags are used for testing. The results are averaged over all four possible combinations. Note that the supervision provided to SSK–SIL requires sig-

[1] http://opennlp.sourceforge.net

[2] http://www.csie.ntu.edu.tw/~cjlin/libsvm



(a) Corporate Acquisitions (b) Person-Birthplace

Figure 3: Precision-Recall graphs on the two datasets.

nificantly more annotation effort, therefore, given a sufficient amount of training examples, we expect this system to perform at least as well as its MIL counterpart.

In Figure 3, precision is plotted against recall by varying a threshold on the value of the SVM decision function. To avoid clutter, we show only the graphs for the first three systems. In Table 4 we show the area under the precision recall curves of all four systems. Overall, the learned relation extractors are able to identify the relationship in novel sentences quite accurately and significantly out-perform a bag-of-words baseline. The new version of the subsequence kernel SSK–T1 is significantly more accurate in the MIL setting than the original subsequence kernel SSK–MIL, and is also competitive with SSK–SIL, which was trained using a reasonable amount of manually labeled sentence examples.

Dataset	SSK–MIL	SSK–T1	BW–MIL	SSK–SIL
(a) CA	76.9%	81.1%	45.9%	80.4%
(b) PB	72.5%	78.2%	69.2%	73.4%

Table 4: Area Under Precision-Recall Curve.

8 Future Work

An interesting potential application of our approach is a web relation-extraction system similar to Google Sets, in which the user provides only a handful of pairs of entities known to exhibit or not to exhibit a particular relation, and the system is used to find other pairs of entities exhibiting the same relation. Ideally, the user would only need to provide positive pairs. Sentences containing one of the relation arguments could be extracted from the web, and likely negative sentence examples automatically created by pairing this entity with other named entities mentioned in the sentence. In this scenario, the training set can contain both false positive and false negative noise. One useful side effect is that Type I bias is partially removed – some bias still remains due to combinations of at least two words, each correlated with a different argument of the relation.

We are also investigating methods for reducing Type II bias, either by modifying the word weights, or by integrating an appropriate measure of word distribution across positive bags directly in the objective function for the MIL problem. Alternatively, implicit negative evidence can be extracted from sentences in positive bags by exploiting the fact that, besides the two relation arguments, a sentence from a positive bag may contain other entity mentions. Any pair of entities different from the relation pair is very likely to be a negative example for that relation. This is similar to the concept of negative *neighborhoods* introduced by Smith and Eisner (2005), and has the potential of eliminating both Type I and Type II bias.

9 Related Work

One of the earliest IE methods designed to work with a reduced amount of supervision is that of Hearst (1992), where a small set of seed patterns is used in a bootstrapping fashion to mine pairs of

hypernym-hyponym nouns. Bootstrapping is actually orthogonal to our method, which could be used as the pattern learner in every bootstrapping iteration. A more recent IE system that works by bootstrapping relation extraction patterns from the web is KNOWITALL (Etzioni et al., 2005). For a given target relation, supervision in KNOWITALL is provided as a *rule template* containing words that describe the class of the arguments (e.g. "company"), and a small set of seed extraction patterns (e.g. "has acquired"). In our approach, the type of supervision is different – we ask only for pairs of entities known to exhibit the target relation or not. Also, KNOWITALL requires large numbers of search engine queries in order to collect and validate extraction patterns, therefore experiments can take weeks to complete. Comparatively, the approach presented in this paper requires only a small number of queries: one query per relation pair, and one query for each relation argument.

Craven and Kumlien (1999) create a noisy training set for the subcellular-localization relation by mining Medline for sentences that contain tuples extracted from relevant medical databases. To our knowledge, this is the first approach that is using a "weakly" labeled dataset for relation extraction. The resulting bags however are very dense in positive examples, and they are also many and small – consequently, the two types of bias are not likely to have significant impact on their system's performance.

10 Conclusion

We have presented a new approach to relation extraction that leverages the vast amount of information available on the web. The new RE system is trained using only a handful of entity pairs known to exhibit and not exhibit the target relationship. We have extended an existing relation extraction kernel to learn in this setting and to resolve problems caused by the minimal supervision provided. Experimental results demonstrate that the new approach can reliably extract relations from web documents.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful suggestions. This work was supported by grant IIS-0325116 from the NSF, and a gift from Google Inc.

References

Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. 2003. Support vector machines for multiple-instance learning. In *NIPS 15*, pages 561–568, Vancouver, BC. MIT Press.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proc. of COLING–ACL '98*, pages 86–90, San Francisco, CA. Morgan Kaufmann Publishers.

Razvan C. Bunescu and Raymond J. Mooney. 2006. Subsequence kernels for relation extraction. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *NIPS 18*.

M. Craven and J. Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *Proc. of ISMB'99*, pages 77–86, Heidelberg, Germany.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proc. of ACL'04*, pages 423–429, Barcelona, Spain, July.

Thomas G. Dietterich, Richard H. Lathrop, and Tomas Lozano-Perez. 1997. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: an experimental study. *Artificial Intelligence*, 165(1):91–134.

T. Gartner, P.A. Flach, A. Kowalczyk, and A.J. Smola. 2002. Multi-instance kernels. In *In Proc. of ICML'02*, pages 179–186, Sydney, Australia, July. Morgan Kaufmann.

M. A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. of ACL'92*, Nantes, France.

Judea Pearl. 1986. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29(3):241–288.

Soumya Ray and Mark Craven. 2005. Supervised versus multiple instance learning: An empirical comparison. In *Proc. of ICML'05*, pages 697–704, Bonn, Germany.

Bernhard Schölkopf and Alexander J. Smola. 2002. *Learning with kernels - support vector machines, regularization, optimization and beyond*. MIT Press, Cambridge, MA.

N. A. Smith and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. of ACL'05*, pages 354–362, Ann Arbor, Michigan.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. John Wiley & Sons.

D. Zelenko, C. Aone, and A. Richardella. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106.

Q. Zhang, S. A. Goldman, W. Yu, and J. Fritts. 2002. Content-based image retrieval using multiple-instance learning. In *Proc. of ICML'02*, pages 682–689.

A Seed-driven Bottom-up Machine Learning Framework for Extracting Relations of Various Complexity

Feiyu Xu, Hans Uszkoreit and Hong Li
Language Technology Lab, DFKI GmbH
Stuhlsatzenhausweg 3, D-66123 Saarbruecken
{feiyu,uszkoreit,hongli}@dfki.de

Abstract

A minimally supervised machine learning framework is described for extracting relations of various complexity. Bootstrapping starts from a small set of n-ary relation instances as "seeds", in order to automatically learn pattern rules from parsed data, which then can extract new instances of the relation and its projections. We propose a novel rule representation enabling the composition of n-ary relation rules on top of the rules for projections of the relation. The compositional approach to rule construction is supported by a bottom-up pattern extraction method. In comparison to other automatic approaches, our rules cannot only localize relation arguments but also assign their exact target argument roles. The method is evaluated in two tasks: the extraction of Nobel Prize awards and management succession events. Performance for the new Nobel Prize task is strong. For the management succession task the results compare favorably with those of existing pattern acquisition approaches.

1 Introduction

Information extraction (IE) has the task to discover n-tuples of relevant items (entities) belonging to an n-ary relation in natural language documents. One of the central goals of the ACE program[1] is to develop a more systematically grounded approach to IE starting from elementary entities, binary relations to n-ary relations such as events. Current semi- or unsupervised approaches to automatic pattern acquisition are either limited to a certain linguistic representation (e.g., subject-verb-object), or only deal with binary relations, or cannot assign slot filler roles to the extracted arguments, or do not have good selection and filtering methods to handle the large number of tree patterns (Riloff, 1996; Agichtein and Gravano, 2000; Yangarber, 2003; Sudo et al., 2003; Greenwood and Stevenson, 2006; Stevenson and Greenwood, 2006). Most of these approaches do not consider the linguistic interaction between relations and their projections on k dimensional subspaces where $1 \leq k < n$, which is important for scalability and reusability of rules. Stevenson and Greenwood (2006) present a systematic investigation of the pattern representation models and point out that substructures of the linguistic representation and the access to the embedded structures are important for obtaining a good coverage of the pattern acquisition. However, all considered representation models (subject-verb-object, chain model, linked chain model and subtree model) are verb-centered. Relations embedded in non-verb constructions such as a compound noun cannot be discovered:

(1) *the 2005 Nobel Peace Prize*

(1) describes a ternary relation referring to three properties of a prize: year, area and prize name. We also observe that the automatically acquired patterns in Riloff (1996), Yangarber (2003), Sudo et al. (2003), Greenwood and Stevenson (2006) cannot be directly used as relation extraction rules because the relation-specific argument role information is missing. E.g., in the management succession domain that concerns the identification of job changing events, a person can either move into a

[1] http://projects.ldc.upenn.edu/ace/

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 584–591,
Prague, Czech Republic, June 2007.

job (called Person_In) or leave a job (called Person_Out). (2) is a simplified example of patterns extracted by these systems:

(2) <*subject: person*> *verb* <*object:organisation*>

In (2), there is no further specification of whether the person entity in the subject position is Person_In or Person_Out.

The ambitious goal of our approach is to provide a general framework for the extraction of relations and events with various complexity. Within this framework, the IE system learns extraction patterns automatically and induces rules of various complexity systematically, starting from sample relation instances as seeds. The arity of the seed determines the complexity of extracted relations. The seed helps us to identify the explicit linguistic expressions containing mentionings of relation instances or instances of their *k*-ary projections where $1 \leq k < n$. Because our seed samples are not linguistic patterns, the learning system is not restricted to a particular linguistic representation and is therefore suitable for various linguistic analysis methods and representation formats. The pattern discovery is bottom-up and compositional, i.e., complex patterns can build on top of simple patterns for projections.

We propose a rule representation that supports this strategy. Therefore, our learning approach is seed-driven and bottom-up. Here we use dependency trees as input for pattern extraction. We consider only trees or their subtrees containing seed arguments. Therefore, our method is much more efficient than the subtree model of Sudo et al., (2003), where all subtrees containing verbs are taken into account. Our pattern rule ranking and filtering method considers two aspects of a pattern: its domain relevance and the trustworthiness of its origin. We tested our framework in two domains: Nobel Prize awards and management succession. Evaluations have been conducted to investigate the performance with respect to the seed parameters: the number of seeds and the influence of data size and its redundancy property. The whole system has been evaluated for the two domains considering precision and recall. We utilize the evaluation strategy "Ideal Matrix" of Agichtein and Gravano (2000) to deal with unannotated test data.

The remainder of the paper is organised as follows: Section 2 provides an overview of the system architecture. Section 3 discusses the rule representation. In Section 4, a detailed description of the seed-driven bottom-up pattern acquisition is presented. Section 5 describes our experiments with pattern ranking, filtering and rule induction. Section 6 presents the experiments and evaluations for the two application domains. Section 7 provides a conclusion and an outline of future work.

2 System Architecture

Given the framework, our system architecture can be depicted as follows:

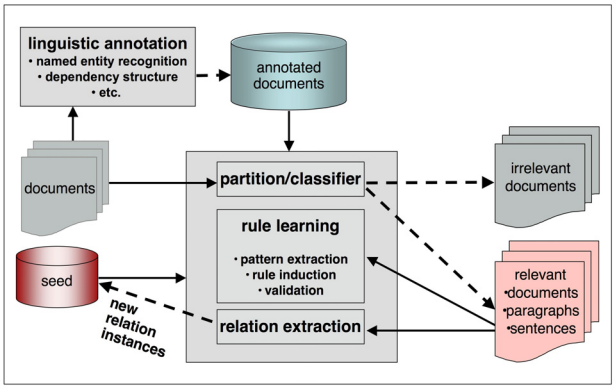


Figure 1. Architecture

This architecture has been inspired by several existing seed-oriented minimally supervised machine learning systems, in particular by Snowball (Agichtein and Gravano, 2000) and ExDisco (Yangarber et al., 2000). We call our system ***DARE***, standing for "***D***omain ***A***daptive ***R***elation ***E***xtraction based on Seeds". ***DARE*** contains four major components: linguistic annotation, classifier, rule learning and relation extraction. The first component only applies once, while the last three components are integrated in a bootstrapping loop. At each iteration, rules will be learned based on the seed and then new relation instances will be extracted by applying the learned rules. The new relation instances are then used as seeds for the next iteration of the learning cycle. The cycle terminates when no new relations can be acquired.

The **linguistic annotation** is responsible for enriching the natural language texts with linguistic information such as named entities and dependency structures. In our framework, the depth of the linguistic annotation can be varied depending on the domain and the available resources.

The **classifier** has the task to deliver relevant paragraphs and sentences that contain seed elements. It has three subcomponents: document re-

trieval, paragraph retrieval and sentence retrieval. The document retrieval component utilizes the open source IR-system Lucene[2]. A translation step is built in to convert the seed into the proper IR query format. As explained in Xu et al. (2006), we generate all possible lexical variants of the seed arguments to boost the retrieval coverage and formulate a boolean query where the arguments are connected via conjunction and the lexical variants are associated via disjunction. However, the translation could be modified. The task of paragraph retrieval is to find text snippets from the relevant documents where the seed relation arguments co-occur. Given the paragraphs, a sentence containing at least two arguments of a seed relation will be regarded as relevant.

As mentioned above, the **rule learning** component constitutes the core of our system. It identifies patterns from the annotated documents inducing extraction rules from the patterns, and validates them. In section 4, we will give a detailed explanation of this component.

The **relation extraction** component applies the newly learned rules to the relevant documents and extracts relation instances. The validated relation instances will then be used as new seeds for the next iteration.

3 *DARE* Rule Representation

Our rule representation is designed to specify the location and the role of the arguments w.r.t. the target relation in a linguistic construction. In our framework, the rules should not be restricted to a particular linguistic representation and should be adaptable to various NLP tools on demand. A ***DARE*** rule is allowed to call further ***DARE*** rules that extract a subset of the arguments. Let us step through some example rules for the prize award domain. One of the target relations in the domain is about a person who obtains a special prize in a certain area in a certain year, namely, a quaternary tuple, see (3). (4) is a domain relevant sentence.

(3) <recipient, prize, area, year>

(4) *Mohamed ElBaradei won the 2005 Nobel Peace Prize on Friday for his efforts to limit the spread of atomic weapons.*

(5) is a rule that extracts a ternary projection instance <*prize, area, year*> from a noun phrase compound, while (6) is a rule which triggers (5) in its object argument and extracts all four arguments. (5) and (6) are useful rules for extracting arguments from (4).

(5)

```
Rule name::  prize_area_year_1
Rule::    (③ year)(① prizename) (② areaname) 'Prize'
Output:: ⟨①Prize, ②Area, ③Year⟩
```

(6)

```
Rule name:: recipient_prize_area_year_1
Rule body:: [ head    [ pos       verb
                        mode      active
                        wordform  "win" ]
              subject [ head  ① Person
                        rule  recipient_1:: ⟨①Person⟩ ]
              object  [ head  [ wordform  "prize" ]
                        rule  prize_area_year_1:: ⟨②Prize, ③Area, ④Year⟩ ] ]
Output:: ⟨①Recipient, ②Prize, ③Area, ④Year⟩
```

Next we provide a definition of a ***DARE*** rule:

A ***DARE*** rule has three components

1. rule name: $\mathbf{r}_i$;
2. output: a set **A** containing the **n** arguments of the n-ary relation**,** labelled with their argument roles;
3. rule body in AVM format containing:
   - specific linguistic labels or attributes (e.g., **subject**, **object**, **head**, **mod**), derived from the linguistic analysis, e.g., dependency structures and the named entity information
   - **rule**: its value is a ***DARE*** rule which extracts a subset of arguments of **A**

The rule in (6) is a typical ***DARE*** rule. Its subject and object descriptions call appropriate ***DARE*** rules that extract a subset of the output relation arguments. The advantages of this rule representation strategy are that (1) it supports the bottom-up rule composition; (2) it is expressive enough for the representation of rules of various complexity; (3) it reflects the precise linguistic relationship among the relation arguments and reduces the template merging task in the later phase; (4) the rules for the subset of arguments may be reused for other relation extraction tasks.

The rule representation models for automatic or unsupervised pattern rule extraction discussed by

[2] http://www.lucene.de

Stevenson and Greenwood (2006) do not account for these considerations.

4 Seed-driven Bottom-up Rule Learning

Two main approaches to seed construction have been discussed in the literature: pattern-oriented (e.g., ExDisco) and semantics-oriented (e.g., Snowball) strategies. The pattern-oriented method suffers from poor coverage because it makes the IE task too dependent on one linguistic representation construction (e.g., *subject-verb-object*) and has moreover ignored the fact that semantic relations and events could be dispersed over different sub-structures of the linguistic representation. In practice, several tuples extracted by different patterns can contribute to one complex relation instance.

The semantics-oriented method uses relation instances as seeds. It can easily be adapted to all relation/event instances. The complexity of the target relation is not restricted by the expressiveness of the seed pattern representation. In Brin (1998) and Agichtein and Gravano (2000), the semantics-oriented methods have proved to be effective in learning patterns for some general binary relations such as *booktitle-author* and *company-headquarter* relations. In Xu et al. (2006), the authors show that at least for the investigated task it is more effective to start with the most complex relation instance, namely, with an n-ary sample for the target n-ary relation as seed, because the seed arguments are often centred in a relevant textual snippet where the relation is mentioned. Given the bottom-up extracted patterns, the task of the rule induction is to cluster and generalize the patterns. In comparison to the bottom-up rule induction strategy (Califf and Mooney, 2004), our method works also in a compositional way. For reasons of space this part of the work will be reported in Xu and Uszkoreit (forthcoming).

4.1 Pattern Extraction

Pattern extraction in ***DARE*** aims to find linguistic patterns which do not only trigger the relations but also locate the relation arguments. In ***DARE***, the patterns can be extracted from a phrase, a clause or a sentence, depending on the location and the distribution of the seed relation arguments.

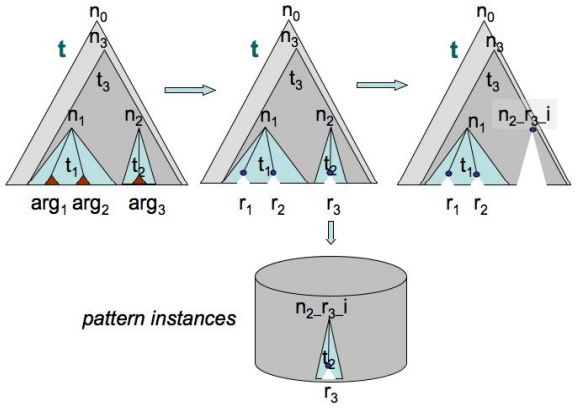


Figure 2. Pattern extraction step 1

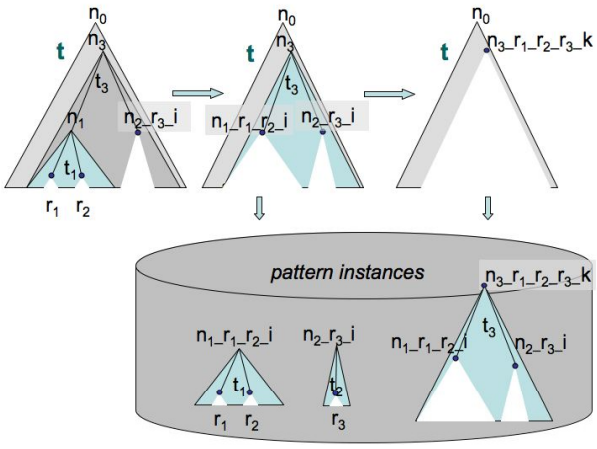


Figure 3. Pattern extraction step 2

Figures 2 and 3 depict the general steps of bottom-up pattern extraction from a dependency tree ***t*** where three seed arguments $\boldsymbol{arg}_1$, $\boldsymbol{arg}_2$ and $\boldsymbol{arg}_3$ are located. All arguments are assigned their relation roles $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ and $\boldsymbol{r}_3$. The pattern-relevant subtrees are trees in which seed arguments are embedded: $\boldsymbol{t}_1$, $\boldsymbol{t}_2$ and $\boldsymbol{t}_3$. Their root nodes are $\boldsymbol{n}_1$, $\boldsymbol{n}_2$ and $\boldsymbol{n}_3$. Figure 2 shows the extraction of a unary pattern $\boldsymbol{n}_2\_\boldsymbol{r}_3\_\boldsymbol{i}$, while Figure 3 illustrates the further extraction and construction of a binary pattern $\boldsymbol{n}_1\_\boldsymbol{r}_1\_\boldsymbol{r}_2\_\boldsymbol{j}$ and a ternary pattern $\boldsymbol{n}_3\_\boldsymbol{r}_1\_\boldsymbol{r}_2\_\boldsymbol{r}_3\_\boldsymbol{k}$. In practice, not all branches in the subtrees will be kept. In the following, we give a general definition of our seed-driven bottom-up pattern extraction algorithm:

input: (i) relation = $<r_1, r_2, ..., r_n>$: the target relation tuple with n argument roles.
T: a set of linguistic analysis trees annotated with i seed relation arguments ($1 \le i \le n$)

output: P: a set of pattern instances which can extract i or a subset of i arguments.

Pattern extraction:
for each tree t $\in$ T

Step 1: (depicted in Figure 2)

1. replace all terminal nodes that are instantiated with the seed arguments by new nodes. Label these new nodes with the seed argument roles and possibly the corresponding entity classes;
2. identify the set of the lowest nonterminal nodes N_1 in t that dominate only one argument (possibly among other nodes).
3. substitute N_1 by nodes labelled with the seed argument roles and their entity classes
4. prune the subtrees dominated by N_1 from t and add these subtrees into P. These subtrees are assigned the argument role information and a unique id.

Step2: For i=2 to n: (depicted in Figure 3)

1. find the set of the lowest nodes N_1 in t that dominate in addition to other children only i seed arguments;
2. substitute N_1 by nodes labelled with the i seed argument role combination information (e.g., $r_i\_r_j$) and with a unique id.
3. prune the subtrees T_i dominated by N_i in t;
4. add T_i to P together with the argument role combination information and the unique id

With this approach, we can learn rules like (6) in a straightforward way.

4.2 Rule Validation: Ranking and Filtering

Our ranking strategy has incorporated the ideas proposed by Riloff (1996), Agichtein and Gravano (2000), Yangarber (2003) and Sudo et al. (2003). We take two properties of a pattern into account:

- *domain relevance*: its distribution in the relevant documents and irrelevant documents (documents in other domains);
- *trustworthiness of its origin*: the relevance score of the seeds from which it is extracted.

In Riloff (1996) and Sudo et al. (2003), the relevance of a pattern is mainly dependent on its occurrences in the relevant documents vs. the whole corpus. Relevant patterns with lower frequencies cannot float to the top. It is known that some complex patterns are relevant even if they have low occurrence rates. We propose a new method for calculating the domain relevance of a pattern. We assume that the domain relevance of a pattern is dependent on the relevance of the lexical terms (words or collocations) constructing the pattern, e.g., the domain relevance of (5) and (6) are dependent on the terms "prize" and "win" respectively. Given *n* different domains, the domain relevance score (*DR*) of a term *t* in a domain d_i is:

$$DR(t, d_i) = \begin{cases} 0, \text{ if } df(t, d_i) = 0; \\ \frac{df(t,d_i)}{N \times D} \times LOG(n \times \frac{df(t,d_i)}{\sum_{j=1}^{n} df(t,d_j)}), \text{ otherwise} \end{cases}$$

where

- *$df(t, d_i)$: is the document frequency of a term t in the domain d_i*
- *D: the number of the documents in d_i*
- *N: the total number of the terms in d_i*

Here the domain relevance of a term is dependent both on its document frequency and its document frequency distribution in other domains. Terms mentioned by more documents within the domain than outside are more relevant (Xu et al., 2002). In the case of *n=3* such different domains might be, e.g., *management succession*, *book review* or *biomedical texts*. Every domain corpus should ideally have the same number of documents and similar average document size. In the calculation of the trustworthiness of the origin, we follow Agichtein and Gravano (2000) and Yangarber (2003). Thus, the relevance of a pattern is dependent on the relevance of its terms and the score value of the most trustworthy seed from which it origins. Finally, the score of a pattern ***p*** is calculated as follows:

$$score(\mathbf{p}) = \sum_{i=0}^{|T|} DR(t_i) \times \max\{score(s) : s \in Seeds\}$$

where $|T| > 0$ and $t_i \in T$

- *T: is the set of the terms occur in p;*
- *Seeds: a set of seeds from which the pattern is extracted;*
- *score(s): is the score of the seed s;*

This relevance score is not dependent on the distribution frequency of a pattern in the domain corpus. Therefore, patterns with lower frequency, in particular, some complex patterns, can be ranked higher when they contain relevant domain terms or come from reliable seeds.

5 Top down Rule Application

After the acquisition of pattern rules, the ***DARE*** system applies these rules to the linguistically annotated corpus. The rule selection strategy moves from complex to simple. It first matches the most complex pattern to the analyzed sentence in order to extract the maximal number of relation arguments. According to the duality principle (Yangarber 2001), the score of the new extracted relation instance *S* is dependent on the patterns from which it origins. Our score method is a simplified version of that defined by Agichtein and Gravano (2000):

$$\text{score}(S)=1-\prod_{i=0}^{|P|}(1-score(P_i))$$

where $P=\{P_i\}$ is the set of patterns that extract S.

The extracted instances can be used as potential seeds for the further pattern extraction iteration, when their scores are validated. The initial seeds obtain 1 as their score.

6 Experiments and Evaluation

We apply our framework to two application domains: Nobel Prize awards and management succession events. Table 1 gives an overview of our test data sets.

Data Set Name	Doc Number	Data Amount
Nobel Prize A (1999-2005)	2296	12,6 MB
Nobel Prize B (1981-1998)	1032	5,8 MB
MUC-6	199	1 MB

Table1. Overview of Test Data Sets.

For the Nobel Prize award scenario, we use two test data sets with different sizes: Nobel Prize A and Nobel Prize B. They are Nobel Prize related articles from New York Times, online BBC and CNN news reports. The target relation for the experiment is a quaternary relation as mentioned in (3), repeated here again:

<recipient, prize, area, year>

Our test data is not annotated with target relation instances. However, the entire list of Nobel Prize award events is available for the evaluation from the Nobel Prize official website[3]. We use it as our reference relation database for building our Ideal table (Agichtein and Gravano, 2000).

For the management succession scenario, we use the test data from MUC-6 (MUC-6, 1995) and define a simpler relation structure than the MUC-6 scenario template with four arguments:

<Person_In, Person_Out, Position, Organisation>

In the following tables, we use PI for Person_In, PO for Person_Out, POS for Position and ORG for Organisation. In our experiments, we attempt to investigate the influence of the size of the seed and the size of the test data on the performance. All these documents are processed by named entity recognition (Drozdzynski et al., 2004) and dependency parser MINIPAR (Lin, 1998).

6.1 Nobel Prize Domain Evaluation

For this domain, three test runs have been evaluated, initialized by one randomly selected relation instance as seed each time. In the first run, we use the largest test data set Nobel Prize A. In the second and third runs, we have compared two random selected seed samples with 50% of the data each, namely Nobel Prize B. For data sets in this domain, we are faced with an evaluation challenge pointed out by DIPRE (Brin, 1998) and Snowball (Agichtein and Gravano, 2000), because there is no gold-standard evaluation corpus available. We have adapted the evaluation method suggested by Agichtein and Gravano, i.e., our system is successful if we capture one mentioning of a Nobel Prize winner event through one instance of the relation tuple or its projections. We constructed two tables (named Ideal) reflecting an approximation of the maximal detectable relation instances: one for Nobel Prize A and another for Nobel Prize B. The Ideal tables contain the Nobel Prize winners that co-occur with the word "Nobel" in the test corpus. Then precision is the correctness of the extracted relation instances, while recall is the coverage of the extracted tuples that match with the Ideal table. In Table 2 we show the precision and recall of the three runs and their random seed sample:

Data Set	Seed	Precision	Recall	
			total	time interval
Nobel Prize A	[Zewail, Ahmed H], nobel, chemistry,1999	**71,6%**	**50,7%**	**70,9%** (1999-2005)
Nobel Prize B	[Sen, Amartya], nobel, economics, 1998	**87,3%**	**31%**	**43%** (1981-1998)
Nobel Prize B	[Arias, Oscar], nobel, peace, 1987	**83,8%**	**32%**	**45%** (1981-1998)

Table 2. Precision, Recall against the Ideal Table

The first experiment with the full test data has achieved much higher recall than the two experiments with the set Nobel Prize B. The two experiments with the Nobel Prize B corpus show similar

3 http://nobelprize.org/

performance. All three experiments have better recalls when taking only the relation instances during the report years into account, because there are more mentionings during these years in the corpus. Figure (6) depicts the pattern learning and new seed extracting behavior during the iterations for the first experiment. Similar behaviours are observed in the other two experiments.

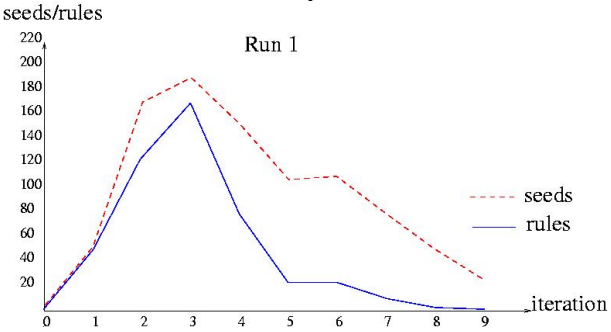


Figure 6. Experiment with Nobel Prize A

6.2 Management Succession Domain

The MUC-6 corpus is much smaller than the Nobel Prize corpus. Since the gold standard of the target relations is available, we use the standard IE precision and recall method. The total gold standard table contains 256 event instances, from which we randomly select seeds for our experiments. Table 3 gives an overview of performance of the experiments. Our tests vary between one seed, 20 seeds and 55 seeds.

Initial Seed Nr.		Precision	Recall
1	A	12.6%	7.0%
	B	15.1%	21.8%
20		48.4%	34.2%
55		62.0%	48.0%

Table 3. Results for various initial seed sets

The first two one-seed tests achieved poor performance. With 55 seeds, we can extract additional 67 instances to obtain the half size of the instances occurring in the corpus. Table 4 show evaluations of the single arguments. B works a little better because the randomly selected single seed appears a better sample for finding the pattern for extracting PI argument.

Arg	precision (A)	precision (B)	Recall (A)	Recall (B)
PI	10.9%	15.1%	8.6%	34.4%
PO	28.6%	–	2.3%	2.3%
ORG	25.6%	100%	2.6%	2.6%
POS	11.2%	11.2%	5.5%	5.5%

Table 4. Evaluation of one-seed tests (A and B)

Table 5 shows the performance with 20 and 55 seeds respectively. Both of them are better than the one-seed tests, while 55 seeds deliver the best performance in average, in particular, the recall value.

arg	precision (20)	precision (55)	recall (20)	recall (55)
PI	84%	62.8%	27.9%	56.1%
PO	41.2%	59%	34.2%	31.2%
ORG	82.4%	58.2%	7.4%	20.2%
POS	42%	64.8%	25.6%	30.6%

Table 5. Evaluation of 20 and 55 seeds tests

Our result with 20 seeds (precision of 48.4% and recall of 34.2%) is comparable with the best result reported by Greenwood and Stevenson (2006) with the linked chain model (precision of 0.434 and recall of 0.265). Since the latter model uses patterns as seeds, applying a similarity measure for pattern ranking, a fair comparison is not possible. Our result is not restricted to binary relations and our model also assigns the exact argument role to the Person role, i.e. Person_In or Person_Out.

We have also evaluated the top 100 event-independent binary relations such as Person-Organisation and Position-Organisation. The precision of these by-product relations of our IE system is above 98%.

7 Conclusion and Future Work

Several parameters are relevant for the success of a seed-based bootstrapping approach to relation extraction. One of these is the arity of the relation. Another one is the locality of the relation instance in an average mentioning. A third one is the types of the relation arguments: Are they named entities in the classical sense? Are they lexically marked? Are there several arguments of the same type? Both tasks we explored involved extracting quaternary relations. The Nobel Prize domain shows better lexical marking because of the prize name. The management succession domain has two slots of the same NE type, i.e., persons. These differences are relevant for any relation extraction approach.

The success of the bootstrapping approach crucially depends on the nature of the training data base. One of the most relevant properties of this data base is the ratio of documents to relation instances. Several independent reports of an instance usually yield a higher number of patterns.

The two tasks we used to investigate our method drastically differ in this respect. The Nobel Prize

domain we selected as a learning domain for general award events since it exhibits a high degree of redundancy in reporting. A Nobel Prize triggers more news reports than most other prizes. The achieved results met our expectations. With one randomly selected seed, we could finally extract most relevant events in some covered time interval.

However, it turns out that it is not just the average number of reports per events that matters but also the distribution of reportings to events. Since the Nobel prizes data exhibit a certain type of skewed distribution, the graph exhibits properties of scale-free graphs. The distances between events are shortened to a few steps. Therefore, we can reach most events in a few iterations. The situation is different for the management succession task where the reports came from a single newspaper. The ratio of events to reports is close to one. This lack of informational redundancy requires a higher number of seeds. When we started the bootstrapping with a single event, the results were rather poor. Going up to twenty seeds, we still did not get the performance we obtain in the Nobel Prize task but our results compare favorably to the performance of existing bootstrapping methods.

The conclusion, we draw from the observed difference between the two tasks is simple: We shall always try to find a highly redundant training data set. If at all possible, the training data should exhibit a skewed distribution of reports to events. Actually, such training data may be the only realistic chance for reaching a large number of rare patterns. In future work we will try to exploit the web as training resource for acquiring patterns while using the parsed domain data as the source for obtaining new seeds in bootstrapping the rules before applying these to any other nonredundant document base. This is possible because our seed tuples can be translated into simple IR queries and further linguistic processing is limited to the retrieved candidate documents.

Acknowledgement

The presented research was partially supported by a grant from the German Federal Ministry of Education and Research to the project Hylap (FKZ: 01IWF02) and EU–funding for the project RASCALLI. Our special thanks go to Doug Appelt and an anonymous reviewer for their thorough and highly valuable comments.

References

E. Agichtein and L. Gravano. 2000. *Snowball: extracting relations from large plain-text collections.* In ACM 2000, pages 85–94, Texas, USA.

S. Brin. *Extracting patterns and relations from the World-Wide Web*. In Proc. 1998 Int'l Workshop on the Web and Databases (WebDB '98), March 1998.

M. E. Califf and R. J. Mooney. 2004. Bottom-Up Relational Learning of Pattern Matching Rules for Information Extraction. *Journal of Machine Learning Research*, MIT Press.

W. Drozdzynski, H.-U.Krieger, J. Piskorski; U. Schäfer, and F. Xu. 2004. Shallow Processing with Unification and Typed Feature Structures — Foundations and Applications. *Künstliche Intelligenz* 1:17—23.

M. A. Greenwood and M. Stevenson. 2006. *Improving Semi-supervised Acquisition of Relation Extraction Patterns*. In Proc. of the Workshop on Information Extraction Beyond the Document, Australia.

D. Lin. 1998. Dependency-based evaluation of MINIPAR. In Workshop on the Evaluation of Parsing Systems, Granada, Spain.

MUC. 1995. Proceedings of the Sixth Message Understanding Conference (MUC-6), Morgan Kaufmann.

E. Riloff. 1996. *Automatically Generating Extraction Patterns from Untagged Text.* In Proc. of the Thirteenth National Conference on Articial Intelligence, pages 1044–1049, Portland, OR, August.

M. Stevenson and Mark A. Greenwood. 2006. *Comparing Information Extraction Pattern Models*. In Proc. of the Workshop on Information Extraction Beyond the Document, Sydney, Australia.

K. Sudo, S. Sekine, and R. Grishman. 2003. *An Improved Extraction Pattern Representation Model for Automatic IE Pattern Acquisition*. In Proc. of ACL-03, pages 224–231, Sapporo, Japan.

R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. 2000. *Automatic Acquisition of Domain Knowledge for Information Extraction*. In Proc. of COLING 2000, Saarbrücken, Germany.

R. Yangarber. 2003. *Counter-training in the Discovery of Semantic Patterns.* In Proceedings of ACL-03, pages 343–350, Sapporo, Japan.

F. Xu, D. Kurz, J. Piskorski and S. Schmeier. 2002. *A Domain Adaptive Approach to Automatic Acquisition of Domain Relevant Terms and their Relations with Bootstrapping*. In Proc. of LREC 2002, May 2002.

F. Xu, H. Uszkoreit and H. Li. 2006. *Automatic Event and Relation Detection with Seeds of Varying Complexity*. In Proceedings of AAAI 2006 Workshop Event Extraction and Synthesis, Boston, July, 2006.

A Multi-resolution Framework for Information Extraction from Free Text

Mstislav Maslennikov and Tat-Seng Chua
Department of Computer Science
National University of Singapore
{maslenni,chuats}@comp.nus.edu.sg

Abstract

Extraction of relations between entities is an important part of Information Extraction on free text. Previous methods are mostly based on statistical correlation and dependency relations between entities. This paper re-examines the problem at the multi-resolution layers of phrase, clause and sentence using dependency and discourse relations. Our multi-resolution framework ARE (Anchor and Relation) uses clausal relations in 2 ways: 1) to filter noisy dependency paths; and 2) to increase reliability of dependency path extraction. The resulting system outperforms the previous approaches by 3%, 7%, 4% on MUC4, MUC6 and ACE RDC domains respectively.

1 Introduction

Information Extraction (IE) is the task of identifying information in texts and converting it into a predefined format. The possible types of information include entities, relations or events. In this paper, we follow the IE tasks as defined by the conferences MUC4, MUC6 and ACE RDC: slot-based extraction, template filling and relation extraction, respectively.

Previous approaches to IE relied on co-occurrence (Xiao et al., 2004) and dependency (Zhang et al., 2006) relations between entities. These relations enable us to make reliable extraction of correct entities/relations at the level of a single clause. However, Maslennikov et al. (2006) reported that the increase of relation path length will lead to considerable decrease in performance. In most cases, this decrease in performance occurs because entities may belong to different clauses. Since clauses in a sentence are connected by clausal relations (Halliday and Hasan, 1976), it is thus important to perform discourse analysis of a sentence.

Discourse analysis may contribute to IE in several ways. First, Taboada and Mann (2005) reported that discourse analysis helps to decompose long sentences into clauses. Therefore, it helps to distinguish relevant clauses from non-relevant ones. Second, Miltsakaki (2003) stated that entities in subordinate clauses are less salient. Third, the knowledge of textual structure helps to interpret the meaning of entities in a text (Grosz and Sidner 1986). As an example, consider the sentences *"ABC Co. appointed a new chairman. Additionally, the current CEO was retired"*. The word 'additionally' connects the event in the second sentence to the entity 'ABC Co.' in the first sentence. Fourth, Moens and De Busser (2002) reported that discourse segments tend to be in a fixed order for structured texts such as court decisions or news. Hence, analysis of discourse order may reduce the variability of possible relations between entities.

To model these factors, we propose a multi-resolution framework ARE that integrates both discourse and dependency relations at 2 levels. ARE aims to filter noisy dependency relations from training and support their evaluation with discourse relations between entities. Additionally, we encode semantic roles of entities in order to utilize semantic relations. Evaluations on MUC4, MUC6 and ACE RDC 2003 corpora demonstrates that our approach outperforms the state-of-art systems mainly due to modeling of discourse relations.

The contribution of this paper is in applying discourse relations to supplement dependency relations in a multi-resolution framework for IE. The

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 592–599,
Prague, Czech Republic, June 2007.

framework enables us to connect entities in different clauses and thus improve the performance on long-distance dependency paths.

Section 2 describes related work, while Section 3 presents our proposed framework, including the extraction of anchor cues and various types of relations, integration of extracted relations, and complexity classification. Section 4 describes our experimental results, with the analysis of results in Section 5. Section 6 concludes the paper.

2 Related work

Recent work in IE focuses on relation-based, semantic parsing-based and discourse-based approaches. Several recent research efforts were based on modeling relations between entities. Culotta and Sorensen (2004) extracted relationships using dependency-based kernel trees in Support Vector Machines (SVM). They achieved an F_1-measure of 63% in relation detection. The authors reported that the primary source of mistakes comes from the heterogeneous nature of non-relation instances. One possible direction to tackle this problem is to carry out further relationship classification. Maslennikov et al. (2006) classified relation path between candidate entities into simple, average and hard cases. This classification is based on the length of connecting path in dependency parse tree. They reported that dependency relations are not reliable for the hard cases, which, in our opinion, need the extraction of discourse relations to supplement dependency relation paths.

Surdeanu et al. (2003) applied semantic parsing to capture the predicate-argument sentence structure. They suggested that semantic parsing is useful to capture verb arguments, which may be connected by long-distance dependency paths. However, current semantic parsers such as the ASSERT are not able to recognize support verb constructions such as *"X conducted an attack on Y"* under the verb frame "attack" (Pradhan et al. 2004). Hence, many useful predicate-argument structures will be missed. Moreover, semantic parsing belongs to the intra-clausal level of sentence analysis, which, as in the dependency case, will need the support of discourse analysis to bridge inter-clausal relations.

Webber et al. (2002) reported that discourse structure helps to extract anaphoric relations. However, their set of grammatical rules is heuristic. Our task needs construction of an automated approach to be portable across several domains. Cimiano et al. (2005) employed a discourse-based analysis for IE. However, their approach requires a predefined domain-dependent ontology in the format of extended logical description grammar as described by Cimiano and Reely (2003). Moreover, they used discourse relations between events, whereas in our approach, discourse relations connect entities.

3 Motivation for using discourse relations

Our method is based on Rhetorical Structure Theory (RST) by Taboada and Mann (2005). RST splits the texts into 2 parts: a) nuclei, the most important parts of texts; and b) satellites, the secondary parts. We can often remove satellites without losing the meaning of text. Both nuclei and satellites are connected with discourse relations in a hierarchical structure. In our work, we use 16 classes of discourse relations between clauses: *Attribution, Background, Cause, Comparison, Condition, Contrast, Elaboration, Enablement, Evaluation, Explanation, Joint, Manner-Means, Topic-Comment, Summary, Temporal, Topic-Change*. The additional 3 relations impose a tree structure: *textual-organization, span* and *same-unit*. All the discourse relation classes are potentially useful, since they encode some knowledge about textual structure. Therefore, we decide to include all of them in the learning process to learn patterns with best possible performance.

We consider two main rationales for utilizing discourse relations to IE. First, discourse relations help to narrow down the search space to the level of a single clause. For example, the sentence *"[<Soc-A1>**Trudeau**</>'s <Soc-A2>**son**</> told everyone], [their prime minister was his father], [who took him to a secret base in the arctic] [and let him peek through a window]."* contains 4 clauses and 7 anchor cues (key phrases) for the type Social, which leads to 21 possible variants. Splitting this sentence into clauses reduces the combinations to 4 possible variants. Additionally, this reduction eliminates the long and noisy dependency paths.

Second, discourse analysis enables us to connect entities in different clauses with clausal relations. As an example, we consider a sentence *"It's a dark comedy about a boy named <AT-A1>**Marshal**</> played by Amourie Kats who discovers all kinds of*

*on and scary things going on in <AT-A2>**a seemingly quiet little town**</>"*. In this example, we need to extract the relation "At" between the entities "Marshal" and "a seemingly quiet little town". The discourse structure of this sentence is given in Figure 1.

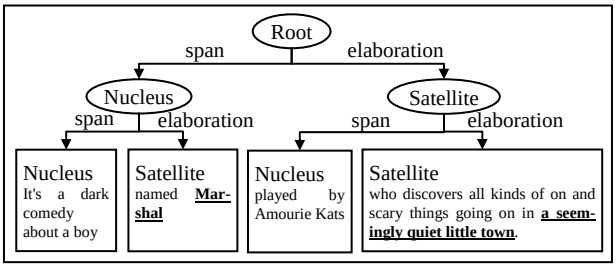


Figure 1. Example of discourse parsing

The discourse path *"Marshal <-elaboration- _ <-span- _ -elaboration-> _ -elaboration-> town"* is relatively short and captures the necessary relations. At the same time, prediction based on dependency path *"Marshal <–obj- _ <-i- _ <-fc- _ <-pnmod- _ <-pred- _ <-i- _ <-null- _ -null-> _ -rel-> _ -i-> _ -mod-> _ -pcomp-n-> town"* is unreliable, since the relation path is long. Thus, it is important to rely on discourse analysis in this example. In addition, we need to evaluate both the score and reliability of prediction by relation path of each type.

4 Anchors and Relations

In this section, we define the key components that we use in ARE: anchors, relation types and general architecture of our system. Some of these components are also presented in detail in our previous work (Maslennikov et al., 2006).

4.1 Anchors

The first task in IE is to identify candidate phrases (which we call anchor or anchor cue) of a predefined type (anchor type) to fill a desired slot in an IE template. The example anchor for the phrase "Marshal" is shown in Figure 2. Given a training set of sentences, we extract the anchor cues $A_{Cj} = [A_1, \ldots, A_{Nanch}]$ of type C_j using the procedures described in Maslennikov et al. (2006). The linguistic features of these anchors for the anchor types of Perpetrator, Action, Victim and Target for the MUC4 domain are given in Table 1.

Figure 2. Example of anchor

Anchor types / Feature	Perpetrator_Cue (A)	Action_Cue (D)	Victim_Cue (A)	Target_Cue (A)
Lexical (Head noun)	terrorists, individuals, Soldiers	attacked, murder, Massacre	Mayor, general, priests	bridge, house, Ministry
Part-of-Speech	Noun	Verb	Noun	Noun
Named Entities	Soldiers (PERSON)	-	Jesuit priests (PERSON)	WTC (OBJECT)
Synonyms	Synset 130, 166	Synset 22	Synset 68	Synset 71
Concept Class	ID 2, 3	ID 9	ID 22, 43	ID 61, 48
Co-referenced entity	He -> terrorist, soldier	-	They -> peasants	-
Clausal type	Nucleus Satellite	Nucleus, Satellite	Nucleus, Satellite	Nucleus, Satellite
Argument type	Arg_0, Arg_1	Target, -, ArgM-MNR	Arg_0, Arg_1	Arg_1, ArgM-MNR

Table 1. Linguistic features for anchor extraction

Given an input phrase P from a test sentence, we need to classify if the phrase belongs to anchor cue type C_j. We calculate the entity score as:

$$Entity\_Score(P) = \sum \delta_i * Feature\_Score_i(P,C_j) \quad (1)$$

where $Feature\_Score(P,C_j)$ is a score function for a particular linguistic feature representation of type C_j, and δ_i is the corresponding weight for that representation in the overall entity score. The weights are learned automatically using Expectation Maximization (Dempster et al., 1977). The $Feature\_Score_i(P,C_j)$ is estimated from the training set as the number of slots containing the correct feature representation type versus all the slots:

$$Feature\_Score_i(P,C_j) = \#(positive\ slots) / \#(all\ slots) \quad (2)$$

We classify the phrase P as belonging to an anchor type C_j when its $Entity\_score(P)$ is above an empirically determined threshold ω. We refer to this anchor as A_j. We allow a phrase to belong to multiple anchor types and hence the anchors alone are not enough for filling templates.

4.2 Relations

To resolve the correct filling of phrase P of type C_i in a desired slot in the template, we need to consider the relations between multiple candidate phrases of related slots. To do so, we consider several types of relations between anchors: discourse, dependency and semantic relations. These relations capture the interactions between anchors and are therefore useful for tackling the paraphrasing and alignment problems (Maslennikov et al., 2006). Given 2 anchors A_i and A_j of anchor types C_i and C_j, we consider a relation $Path_l = [A_i, Rel_1, \ldots, Rel_n, A_j]$ between them, such that there are no anchors between A_i and A_j. Additionally, we assume that the relations between anchors are represented in the form of a tree T_l, where $l = \{s, c, d\}$ refers to

discourse, dependency and semantic relation types respectively. We describe the nodes and edges of T_l separately for each type, because their representations are different:

1) The nodes of discourse tree T_c consist of clauses $[Clause_1, ..., Clause_{Ncl}]$; and their relation edges are obtained from the Spade system described in Soricut and Marcu (2003). This system performs RST-based parsing at the sentence level. The reported accuracy of Spade is 49% on the RST-DT corpus. To obtain a clausal path, we map each anchor A_i to its clause in Spade. If anchors A_i and A_j belong to the same clause, we assign them the relation *same-clause*.
2) The nodes of dependency tree T_d consist of words in sentences; and their relation edges are obtained from Minipar by Lin (1997). Lin (1997) reported a parsing performance of Precision = 88.5% and Recall = 78.6% on the SUSANNE corpus.
3) The nodes of semantic tree T_s consist of arguments $[Arg_0, ..., Arg_{Narg}]$ and targets $[Target_1, ..., Target_{Ntarg}]$. Both arguments and targets are obtained from the ASSERT parser developed by Pradhan (2004). The reported performance of ASSERT is F_1=83.8% on the identification and classification task for all arguments, evaluated using PropBank and AQUAINT as the training and testing corpora, respectively. Since the relation edges have a form $Target_k$ -> Arg_l, the relation path in semantic frame contains only a single relation. Therefore, we encode semantic relations as part of the anchor features.

In later parts of this paper, we consider only discourse and dependency relation paths $Path_l$, where $l=\{c, d\}$.

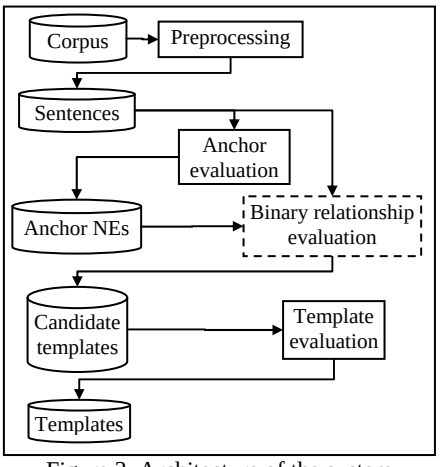


Figure 3. Architecture of the system

4.3 Architecture of ARE system

In order to perform IE, it is important to extract candidate entities (anchors) of appropriate anchor types, evaluate the relationships between them, further evaluate all possible candidate templates, and output the final template. For the case of relation extraction task, the final templates are the same as an extracted binary relation. The overall architecture of ARE is given in Figure 3.

The focus of this paper is in applying discourse relations for binary relationship evaluation.

5 Overall approach

In this section, we describe our relation-based approach to IE. We start with the evaluation of relation paths (single relation ranking, relation path ranking) to assess the suitability of their anchors as entities to template slots. Here we want to evaluate given a single relation or relation path, whether the two anchors are correct in filling the appropriate slots in a template. This is followed by the integration of relation paths and evaluation of templates.

5.1 Evaluation of relation path

In the first stage, we evaluate from training data the relevance of relation path $Path_l = [A_i, Rel_1, ..., Rel_n, A_j]$ between candidate anchors A_i and A_j of types C_i and C_j. We divide this task into 2 steps. The first step ranks each single relation $Rel_k \in Path_l$; while the second step combines the evaluations of Rel_k to rank the whole relation path $Path_l$.

Single relation ranking

Let Set_i and Set_j be the set of linguistic features of anchors A_i and A_j respectively. To evaluate Rel_k, we consider 2 characteristics: (1) the direction of relation Rel_k as encoded in the tree structure; and (2) the linguistic features, Set_i and Set_j, of anchors A_i and A_j. We need to construct multiple single relation classifiers, one for each anchor pair of types C_i and C_j, to evaluate the relevance of Rel_k with respect to these 2 anchor types.

(a) Construction of classifiers. The training data to each classifier consists of anchor pairs of types C_i and C_j extracted from the training corpus. We use these anchor pairs to construct each classifier in four stages. First, we compose the set of possible patterns in the form $P^+ = \{ P_m = <S_i –Rel-> S_j> \mid S_i \in Set_i , S_j \in Set_j \}$. The construction of P_m

conforms to the 2 characteristics given above. Figure 4 illustrates several discourse and dependency patterns of P^+ constructed from a sample sentence.

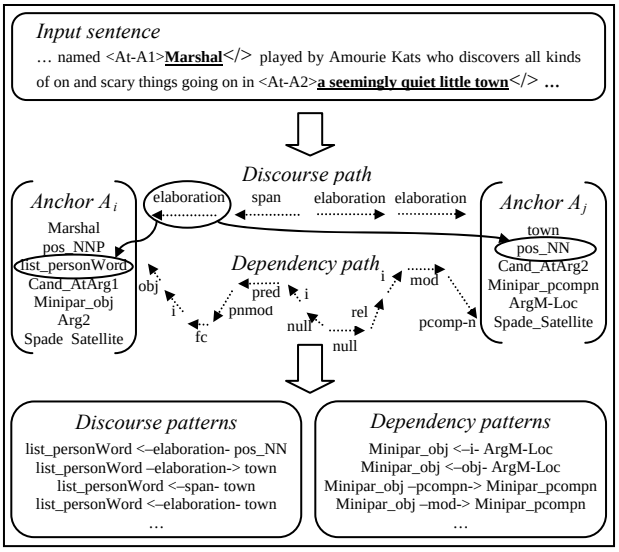


Figure 4. Examples of discourse and dependency patterns

Second, we identify the candidate anchor A, whose type matches slot C in a template. Third, we find the correct patterns for the following 2 cases: 1) A_i, A_j are of correct anchor types; and 2) A_i is an action anchor, while A_j is a correct anchor. Any other patterns are considered as incorrect. We note that the discourse and dependency paths between anchors A_i and A_j are either correct or wrong simultaneously.

Fourth, we evaluate the relevance of each pattern $P_m \in P^+$. Given the training set, let $PairSet_m$ be the set of anchor pairs extracted by P_m; and $PairSet^+(C_i, C_j)$ be the set of correct anchor pairs of types C_i, C_j. We evaluate both precision and recall of P_m as

$$Precision(P_m) = \frac{\| PairSet_m \text{ I } PairsSet^+(C_i, C_j) \|}{\| PairSet_m \|} \quad (3)$$

$$Recall(P_m) = \frac{\| PairSet_m \text{ I } PairsSet^+(C_i, C_j) \|}{\| PairsSet^+(C_i, C_j) \|} \quad (4)$$

These values are stored and used in the training model for use during testing.

(b) Evaluation of relation. Here we want to evaluate whether relation *InputRel* belongs to a path between anchors $InputA_i$ and $InputA_j$. We employ the constructed classifier for the anchor types $InputC_i$ and $InputC_j$ in 2 stages. First, we find a subset $P^{(0)} = \{ P_m = <S_i \text{ –}InputRel\text{->} S_j> \in P^+ \mid S_i \in InputSet_i, S_j \in InputSet_j \}$ of applicable patterns. Second, we utilize $P^{(0)}$ to find the pattern $P_m^{(0)}$ with maximal precision:

$$Precision(P_m^{(0)}) = argmax_{Pm \in P(0)} Precision(P_m) \quad (5)$$

A problem arises if $P_m^{(0)}$ is evaluated only on a small amount of training instances. For example, we noticed that patterns that cover 1 or 2 instances may lead to $Precision=1$, whereas on the testing corpus their accuracy becomes less than 50%. Therefore, it is important to additionally consider the recall parameter of $P_m^{(0)}$.

Relation path ranking

In this section, we want to evaluate relation path connecting template slots C_i and C_j. We do this independently for each relation of type discourse and dependency. Let $Recall_k$ and $Precision_k$ be the recall and precision values of Rel_k in $Path = [A_i, Rel_1, \ldots, Rel_n, A_j]$, both obtained from the previous step. First, we calculate the average recall of the involved relations:

$$W = (1/Length_{Path}) * \Sigma_{Relk \in Path} Recall_k \quad (6)$$

W gives the average recall of the involved relations and can be used as a measure of reliability of the relation *Path*. Next, we compute a combined score of average $Precision_k$ weighted by $Recall_k$:

$$Score = 1/(W*Length_{Path}) * \Sigma_{Relk \in Path} Recall_k * Precision_k \quad (7)$$

We use all $Precision_k$ values in the path here, because omitting a single relation may turn a correct path into the wrong one, or vice versa. The combined score value is used as a ranking of the relation path. Experiments show that we need to give priority to scores with higher reliability W. Hence we use $(W, Score)$ to evaluate each *Path*.

5.2 Integration of different relation path types

The purpose of this stage is to integrate the evaluations for different types of relation paths. The input to this stage consists of evaluated relation paths $Path_C$ and $Path_D$ for discourse and dependency relations respectively. Let $(W_l, Score_l)$ be an evaluation for $Path_l$, $l \in [c, d]$. We first define an integral path $Path_I$ between A_i and A_j as: 1) $Path_I$ is enabled if at least one of $Path_l$, $l \in [c, d]$, is enabled; and 2) $Path_I$ is correct if at least one of $Path_l$ is correct. To evaluate $Path_I$, we consider the average recall W_l of each $Path_l$, because W_l esti-

mates the reliability of $Score_I$. We define a weighted average for $Path_I$ as:

$$W_I = W_C + W_D \quad (8)$$

$$Score_I = 1/W_I * \sum_l W_l * Score_l \quad (9)$$

Next, we want to determine the threshold score $Score_I^O$ above which $Score_I$ is acceptable. This score may be found by analyzing the integral paths on the training corpus. Let $S_I = \{ Path_I \}$ be the set of integral paths between anchors A_i and A_j on the training set. Among the paths in S_I, we need to define a set function $S_I(X) = \{ Path_I \mid Score_I(Path_I) \geq X \}$ and find the optimal threshold for X. We find the optimal threshold based on F_1-measure, because precision and recall are equally important in IE. Let $S_I(X)^+ \subset S_I(X)$ and $S(X)^+ \subset S(X)$ be sets of correct path extractions. Let $F_I(X)$ be F_1-measure of $S_I(X)$:

$$P_I(X) = \frac{\| S_I(X)^+ \|}{\| S_I(X) \|} \quad (10) \qquad R_I(X) = \frac{\| S_I(X)^+ \|}{\| S(X)^+ \|} \quad (11)$$

$$F_I(X) = \frac{2 * P_I(X) * R_I(X)}{P_I(X) + R_I(X)} \quad (12)$$

Based on the computed values $F_I(X)$ for each X on the training data, we determine the optimal threshold as $Score_I^O = argmax_X\ F_I(X)$, which corresponds to the maximal expected F_1-measure of anchor pair A_i and A_j.

5.3 Evaluation of templates

At this stage, we have a set of accepted integral relation paths between any anchor pair A_i and A_j. The next task is to merge appropriate set of anchors into candidate templates. Here we follow the methodology of Maslennikov et al. (2006). For each sentence, we compose a set of candidate templates T using the extracted relation paths between each A_i and A_j. To evaluate each template $T_i \in T$, we combine the integral scores from relation paths between its anchors A_i and A_j into the overall $Relation\_Score_T$:

$$Relation\_Score_T(T_i) = \frac{\sum_{1 \leq i,j \leq K} Score_I(A_i, A_j)}{M} \quad (13)$$

where K is the number of extracted slots, M is the number of extracted relation paths between anchors A_i and A_j, and $Score_I(A_i, A_j)$ is obtained from Equation (9).

Next, we calculate the extracted entity score based on the scores of all the anchors in T_i:

$$Entity\_Score_T(T_i) = \sum_{1 \leq k \leq K} Entity\_Score(A_k) / K \quad (14)$$

where $Entity\_Score(A_i)$ is taken from Equation (1).

Finally, we obtain the combined evaluation for a template:

$$Score_T(T_i) = (1 - \lambda) * Entity\_Score_T(T_i) + \lambda * Relation\_Score_T(T_i) \quad (15)$$

where λ is a predefined constant.

In order to decide whether the template T_i should be accepted or rejected, we need to determine a threshold $Score_T^O$ from the training data. If anchors of a candidate template match slots in a correct template, we consider the candidate template as correct. Let $TrainT = \{ T_i \}$ be the set of candidate templates extracted from the training data, $TrainT^+ \subset TrainT$ be the subset of correct candidate templates, and $TotalT^+$ be the total set of correct templates in the training data. Also, let $TrainT(X) = \{ T_i \mid Score_T(T_i) \geq X, T_i \in TrainT \}$ be the set of candidate templates with score above X and $TrainT^+(X) \subset TrainT(X)$ be the subset of correct candidate templates. We define the measures of precision, recall and F_1 as follows:

$$P_T(X) = \frac{\| TrainT^+(X) \|}{\| TrainT(X) \|} \quad (16) \qquad R_T(X) = \frac{\| TrainT^+(X) \|}{\| TotalT^+ \|} \quad (17)$$

$$F_T(X) = \frac{2 * P_T(X) R_T(X)}{P_T(X) + R_T(X)} \quad (18)$$

Since the performance in IE is measured in F_1-measure, an appropriate threshold to be used for the most prominent candidate templates is:

$$Score_T^O = argmax_X\ F_T(X) \quad (19)$$

The value $Score_T^O$ is used as a training model. During testing, we accept a candidate template $InputT_i$ if $Score_T(InputT_i) > Score_T^O$.

As an additional remark, we note that domains MUC4, MUC6 and ACE RDC 2003 are significantly different in the evaluation methodology for the candidate templates. While the performance of the MUC4 domain is measured for each slot individually; the MUC6 task measures the performance on the extracted templates; and the ACE RDC 2003 task evaluates performance on the matching relations. To overcome these differences, we construct candidate templates for all the domains and measure the required type of performance for each domain. Our candidate templates for the ACE RDC 2003 task consist of only 2 slots, which correspond to entities of the correct relations.

6 Experimental results

We carry out our experiments on 3 domains: MUC4 (Terrorism), MUC6 (Management Succession), and ACE-Relation Detection and Characterization (2003). The MUC4 corpus contains 1,300 documents as training set and 200 documents (TST3 and TST4) as official testing set. We used a modified version of the MUC6 corpus described by Soderland (1999). This version includes 599 documents as training set and 100 documents as testing set. Following the methodology of Zhang et al. (2006), we use only the English portion of ACE RDC 2003 training data. We used 97 documents for testing and the remaining 155 documents for training. Our task is to extract 5 major relation types and 24 subtypes.

Case (%)	P	R	F_1
GRID	52%	62%	57%
Riloff'05	46%	51%	48%
ARE (2006)	58%	61%	60%
ARE	65%	61%	63%

Table 2. Results on MUC4

To compare the results on the terrorism domain in MUC4, we choose the recent state-of-art systems GRID by Xiao et al. (2004), Riloff et al. (2005) and ARE (2006) by Maslennikov et al. (2006) which does not utilize discourse and semantic relations. The comparative results are given in Table 2. It shows that our enhanced ARE results in 3% improvement in F_1 measure over ARE (2006) that does not use clausal relations. The improvement was due to the use of discourse relations on long paths, such as "X *distributed leaflets claiming responsibility for* murder *of* Y". At the same time, for many instances, it would be useful to store the extracted anchors for another round of learning. For example, the extracted features of discourse pattern "murder –same_clause-> HUM_PERSON" may boost the score for patterns that correspond to relation path "X <-span- _ -Elaboration-> murder". In this way, high-precision patterns will support the refinement of patterns with average recall and low precision. This observation is similar to that described in Ciravegna's work on $(LP)^2$ (Ciravegna 2001).

Case (%)	P	R	F_1
Chieu et al.'02	75%	49%	59%
ARE (2006)	73%	58%	65%
ARE	73%	70%	72%

Table 3. Results on MUC6

Next, we present the performance of our system on MUC6 corpus (Management Succession) as shown in Table 3. The improvement of 7% in F_1 is mainly due to the filtering of irrelevant dependency relations. Additionally, we noticed that 22% of testing sentences contain 2 answer templates, and entities in many of such templates are intertwined. One example is the sentence *"Mr. Bronczek who is 39 years old succeeds Kenneth Newell 55 who was named to the new post of senior vice president"*, which refers to 2 positions. We therefore we need to extract 2 templates *"PersonIn: Bronczek, PersonOut: Newell"* and *"PersonIn: Newell, Post: senior vice president"*. The discourse analysis is useful to extract the second template, while rejecting another long-distance template *"PersonIn: Bronczek, PersonOut: Newell, Post: seniour vice president"*. Another remark is that it is important to assign 2 anchors of 'Cand_PersonIn' and 'Cand_PersonOut' for the phrase "Kenneth Newell".

The characteristic of the ACE corpus is that it contains a large amount of variations, while only 2% of possible dependency paths are correct. Since many of the relations occur only at the level of single clause (for example, most instances of relation *At*), the discourse analysis is used to eliminate long-distance dependency paths. It allows us to significantly decrease the dimensionality of the problem. We noticed that 38% of relation paths in ACE contain a single relation, 28% contain 2 relations and 34% contain ≥ 3 relations. For the case of ≥ 3 relations, the analysis of dependency paths alone is not sufficient to eliminate the unreliable paths. Our results for general types and specific subtypes are presented in Tables 6 and 7, respectively.

Case (%)	P	R	F_1
Zhang et al.'06	77%	65%	70%
ARE	79%	66%	73%

Table 4. Results on ACE RDC'03, general types

Based on our results in Table 4, discourse and dependency relations support each other in different situations. We also notice that multiple instances require modeling of entities in the path. Thus, in our future work we need to enrich the search space for relation patterns. This observation corresponds to that reported in Zhang et al. (2006).

Discourse parsing is very important to reduce the amount of variations for specific types on ACE

RDC'03, as there are 48 possible anchor types.

Case (%)	P	R	F_1
Zhang et al.'06	64%	51%	57%
ARE	67%	54%	61%

Table 5. Results on ACE RDC'03, specific types

The relatively small improvement of results in Table 5 may be attributed to the following reasons: 1) it is important to model the commonality relations, as was done by Zhou et al. (2006); and 2) our relation paths do not encode entities. This is different from Zhang et al. (2006), who were using entities in their subtrees.

Overall, the results indicate that the use of discourse relations leads to improvement over the state-of-art systems.

7 Conclusion

We presented a framework that permits the integration of discourse relations with dependency relations. Different from previous works, we tried to use the information about sentence structure based on discourse analysis. Consequently, our system improves the performance in comparison with the state-of-art IE systems. Another advantage of our approach is in using domain-independent parsers and features. Therefore, ARE may be easily portable into new domains.

Currently, we explored only 2 types of relation paths: dependency and discourse. For future research, we plan to integrate more relations in our multi-resolution framework.

References

P. Cimiano and U. Reyle. 2003. Ontology-based semantic construction, underspecification and disambiguation. *In Proc of the Prospects and Advances in the Syntax-Semantic Interface Workshop*.

P. Cimiano, U. Reyle and J. Saric. 2005. Ontology-driven discourse analysis for information extraction. *Data & Knowledge Engineering*, 55(1):59-83.

H.L. Chieu and H.T. Ng. 2002. A Maximum Entropy Approach to Information Extraction from Semi-Structured and Free Text. *In Proc of AAAI-2002*.

F. Ciravegna. 2001. Adaptive Information Extraction from Text by Rule Induction and Generalization. *In Proc of IJCAI-2001*.

A. Culotta and J. Sorensen J. 2004. Dependency tree kernels for relation extraction. *In Proc of ACL-2004*.

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

B. Grosz and C. Sidner. 1986. Attention, Intentions and the Structure of Discourse. *Computational Linguistics*, 12(3):175-204.

M. Halliday and R. Hasan. 1976. *Cohesion in English*. Longman, London.

D. Lin. 1997. Dependency-based Evaluation of Minipar. *In Workshop on the Evaluation of Parsing systems*.

M. Maslennikov, H.K. Goh and T.S. Chua. 2006. ARE: Instance Splitting Strategies for Dependency Relation-based Information Extraction. *In Proc of ACL-2006*.

E. Miltsakaki. 2003. *The Syntax-Discourse Interface: Effects of the Main-Subordinate Distinction on Attention Structure*. PhD thesis.

M.F. Moens and R. De Busser. 2002. First steps in building a model for the retrieval of court decisions. International *Journal of Human-Computer Studies*, 57(5):429-446.

S. Pradhan, W. Ward, K. Hacioglu, J. Martin and D. Jurafsky. 2004. Shallow Semantic Parsing using Support Vector Machines. *In Proc of HLT/NAACL-2004*.

E. Riloff, J. Wiebe, and W. Phillips. 2005. Exploiting Subjectivity Classification to Improve Information Extraction. *In Proc of AAAI-2005*.

S. Soderland. 1999. Learning Information Extraction Rules for Semi-Structured and Free Text. *Machine Learning*, 34:233-272.

R. Soricut and D. Marcu. 2003. Sentence Level Discourse Parsing using Syntactic and Lexical Information. *In Proc of HLT/NAACL*.

M. Surdeanu, S. Harabagiu, J. Williams, P. Aarseth. 2003. Using Predicate Arguments Structures for Information Extraction. *In Proc of ACL-2003*.

M. Taboada and W. Mann. 2005. Applications of Rhetorical Structure Theory. *Discourse studies*, 8(4).

B. Webber, M. Stone, A. Joshi and A. Knott. 2002. Anaphora and Discourse Structure. *Computational Linguistics*, 29(4).

J. Xiao, T.S. Chua and H. Cui. 2004. Cascading Use of Soft and Hard Matching Pattern Rules for Weakly Supervised Information Extraction. *In Proc of COLING-2004*.

M. Zhang, J. Zhang, J. Su and G. Zhou. 2006. A Composite Kernel to Extract Relations between Entities with both Flat and Structured Features. *In Proc of ACL-2006*.

G. Zhou, J. Su and M. Zhang. 2006. Modeling Commonality among Related Classes in Relation Extraction. *In Proc of ACL-2006*.

Using Corpus Statistics on Entities to Improve Semi-supervised Relation Extraction from the Web

Benjamin Rosenfeld
Information Systems
HU School of Business,
Hebrew University, Jerusalem, Israel
grurgrur@gmail.com

Ronen Feldman
Information Systems
HU School of Business,
Hebrew University, Jerusalem, Israel
ronen.feldman@huji.ac.il

Abstract

Many errors produced by unsupervised and semi-supervised relation extraction (RE) systems occur because of wrong recognition of entities that participate in the relations. This is especially true for systems that do not use separate named-entity recognition components, instead relying on general-purpose shallow parsing. Such systems have greater applicability, because they are able to extract relations that contain attributes of unknown types. However, this generality comes with the cost in accuracy. In this paper we show how to use corpus statistics to validate and correct the arguments of extracted relation instances, improving the overall RE performance. We test the methods on SRES – a self-supervised Web relation extraction system. We also compare the performance of corpus-based methods to the performance of validation and correction methods based on supervised NER components.

1 Introduction

Information Extraction (IE) is the task of extracting factual assertions from text. Most IE systems rely on knowledge engineering or on machine learning to generate the "task model" that is subsequently used for extracting instances of entities and relations from new text. In the knowledge engineering approach the model (usually in the form of extraction rules) is created manually, and in the machine learning approach the model is learned automatically from a manually labeled training set of documents. Both approaches require substantial human effort, particularly when applied to the broad range of documents, entities, and relations on the Web. In order to minimize the manual effort necessary to build Web IE systems, semi-supervised and completely unsupervised systems are being developed by many researchers.

The task of extracting facts from the Web has significantly different aims than the regular information extraction. The goal of regular IE is to identify and label all mentions of all instances of the given relation type inside a document or inside a collection of documents. Whereas, in the Web Extraction (WE) tasks we are only interested in extracting relation instances and not interested in particular mentions.

This difference in goals leads to a difference in the methods of performance evaluation. The usual measures of performance of regular IE systems are precision, recall, and their combinations – the breakeven point and F-measure. Unfortunately, the true recall usually cannot be known for WE tasks. Consequently, for evaluating the performance of WE systems, the recall is substituted by the number of extracted instances.

WE systems usually order the extracted instances by the system's confidence in their correctness. The precision of top-confidence extractions is usually very high, but it gets progressively lower when lower-confidence candidates are considered. The curve that plots the number of extractions against precision level is the best indicator of system's quality. Naturally, for a comparision be-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 600–607,
Prague, Czech Republic, June 2007.

tween different systems to be meaningful, the evaluations must be performed on the same corpus.

In this paper we are concerned with Web RE systems that extract binary relations between named entities. Most of such systems utilize separate named entity recognition (NER) components, which are usualy trained in a supervised way on a separate set of manually labeled documents. The NER components recognize and extract the values of relation attributes (also called *arguments*, or *slots*), while the RE systems are concerned with patterns of contexts in which the slots appear. However, good NER components only exist for common and very general entity types, such as *Person*, *Organization*, and *Location*. For some relations, the types of attributes are less common, and no ready NER components (or ready labeled training sets) exist for them. Also, some Web RE systems (e.g., KnowItAll (Etzioni, Cafarella et al. 2005)) do not use separate NER components even for known entity types, because such components are usually domain-specific and may perform poorly on cross-domain text collections extracted from the Web.

In such cases, the values for relation attributes must be extracted by generic methods – shallow parsing (extracting noun phrases), or even simple substring extraction. Such methods are naturally much less precise and produce many entity-recognition errors (Feldman and Rosenfeld 2006).

In this paper we propose several methods of using corpus statistics to improve Web RE precision by validating and correcting the entities extracted by generic methods. The task of Web Extraction is particularly suited for the corpus statistics-based methods because of very large size of the corpora involved, and because the system is not required to identify individual mentions of the relations.

Our methods of entity validation and correction are based on the following two observations:

First, the entities that appear in target relations will often also appear in many other contexts, some of which may strongly discriminate in favor of entities of specific type. For example, assume the system encounters a sentence "*Oracle bought PeopleSoft.*" If the system works without a NER component, it only knows that "*Oracle*" and "*PeopleSoft*" are proper noun phrases, and its confidence in correctness of a candidate relation instance *Acquisition*(*Oracle*, *PeopleSoft*) cannot be very high. However, both entities occur many times elsewhere in the corpus, sometimes in strongly discriminating contexts, such as "*Oracle is a company that…*" or "*PeopleSoft Inc.*" If the system somehow learned that such contexts indicate entities of the correct type for the *Acquisition* relation (i.e., companies), then the system would be able to boost its confidence in both entities ("*Oracle*" and "*PeopleSoft*") being of correct types and, consequently, in (*Oracle*, *PeopleSoft*) being a correct instance of the *Acquisition* relation.

Another observation that we can use is the fact that the entities, in which we are interested, usually have sufficient frequency in the corpus for statistical term extraction methods to perform reasonably well. These methods may often correct a wrongly placed entity boundary, which is a common mistake of general-purpose shallow parsers.

In this paper we show how to use these observations to supplement a Web RE system with an entity validation and correction component, which is able to significantly improve the system's accuracy. We evaluate the methods using SRES (Feldman and Rosenfeld 2006) – a Web RE system, designed to extend and improve KnowItAll (Etzioni, Cafarella et al. 2005). The contributions of this paper are as follows:

- We show how to automatically generate the validating patterns for the target relation arguments, and how to integrate the results produced by the validating patterns into the whole relation extraction system.
- We show how to use corpus statistics and term extraction methods to correct the boundaries of relation arguments.
- We experimentally compare the improvement produced by the corpus-based entity validation and correction methods with the improvements produced by two alternative validators – a CRF-based NER system trained on a separate labeled corpus, and a small manually-built rule-based NER component.

The rest of the paper is organized as follows: Section 2 describes previous work. Section 3 outlines the general design principles of SRES and briefly describes its components. Section 4 describes in detail the different entity validation and correction methods, and Section 5 presents their

experimental evaluation. Section 6 contains conclusions and directions for future work.

2 Related Work

We are not aware of any work that deals specifically with validation and/or correction of entity recognition for the purposes of improving relation extraction accuracy. However, the background techniques of our methods are relatively simple and known. The validation is based on the same ideas that underlie semi-supervised entity extraction (Etzioni, Cafarella et al. 2005), and uses a simplified SRES code. The boundary correction process utilizes well-known term extraction methods, e.g., (Su, Wu et al. 1994).

We also recently became aware of the work by Downey, Broadhead and Etzioni (2007) that deals with locating entities of arbitrary types in large corpora using corpus statistics.

The IE systems most similar to SRES are based on bootstrap learning: Mutual Bootstrapping (Riloff and Jones 1999), the DIPRE system (Brin 1998), and the Snowball system (Agichtein and Gravano 2000). Ravichandran and Hovy (Ravichandran and Hovy 2002) also use bootstrapping, and learn simple surface patterns for extracting binary relations from the Web.

Unlike these systems, SRES surface patterns allow gaps that can be matched by any sequences of tokens. This makes SRES patterns more general, and allows to recognize instances in sentences inaccessible to the simple surface patterns of systems such as (Brin 1998; Riloff and Jones 1999; Ravichandran and Hovy 2002).

Another direction for unsupervised relation learning was taken in (Hasegawa, Sekine et al. 2004; Chen, Ji et al. 2005). These systems use a NER system to identify frequent pairs of entities and then cluster the pairs based on the types of the entities and the words appearing between the entities. The main benefit of this approach is that all relations between two entity types can be discovered simultaneously and there is no need for the user to supply the relations definitions.

3 Description of SRES

The goal of SRES is extracting instances of specified relations from the Web without human supervision. Accordingly, the supervised input to the system is limited to the specifications of the target relations. A specification for a given relation consists of the *relation schema* and a small set of *seeds* – known true instances of the relation. In the full-scale SRES, the seeds are also generated automatically, by using a set of generic patterns instantiated with the relation schema. However, the seed generation is not relevant to this paper.

A relation schema specifies the name of the relation, the names and types of its arguments, and the arguments ordering. For example, the schema of the *Acquisition* relation

Acquisition(Buyer=ProperNP,
Acquired=ProperNP) ordered

specifies that *Acquisition* has two slots, named *Buyer* and *Acquired*, which must be filled with entities of type *ProperNP*. The order of the slots is important (as signified by the word "*ordered*", and as opposed to relations like *Merger*, which are "*unordered*" or, in binary case, "*symmetric*").

The baseline SRES does not utilize a named entity recognizer, instead using a shallow parser for exracting the relation slots. Thus, the only allowed entity types are *ProperNP*, *CommonNP*, and *AnyNP*, which mean the heads of, respectively, proper, common, and arbitrary noun phrases. In the experimental section we compare the baseline SRES to its extensions containing additional NER components. When using those components we allow further subtypes of *ProperNP*, and the relation schema above becomes

... (Buyer=Company, Acquired=Company) ...

The main components of SRES are the Pattern Learner, the Instance Extractor, and the Classifier. The Pattern Learner uses the seeds to learn likely patterns of relation occurrences. Then, the Instance Extractor uses the patterns to extract the candidate instances from the sentences. Finally, the Classifier assigns the confidence score to each extraction. We shall now briefly describe these components.

3.1 Pattern Learner

The *Pattern Learner* receives a relation schema and a set of seeds. Then it finds the occurences of seeds inside a large (unlabeled) text corpus, analyzes their contexts, and extracts common patterns among these contexts. The details of the patterns language and the process of pattern learning are not significant for this paper, and are described fully in (Feldman and Rosenfeld 2006).

3.2 Instance Extractor

The Instance Extractor applies the patterns generated by the Pattern Learner to the text corpus. In order to be able to match the slots of the patterns, the Instance Extractor utilizes an external shallow parser from the OpenNLP package (http://opennlp.sourceforge.net/), which is able to find all proper and common noun phrases in a sentence. These phrases are matched to the slots of the patterns. In other respects, the pattern matching and extraction process is straightforward.

3.3 Classifier

The goal of the final classification stage is to filter the list of all extracted instances, keeping the correct extractions, and removing mistakes that would always occur regardless of the quality of the patterns. It is of course impossible to know which extractions are correct, but there exist properties of patterns and pattern matches that increase or decrease the confidence in the extractions that they produce.

These properties are turned into a set of binary features, which are processed by a linear feature-rich classifier. The classifier receives a feature vector for a candidate, and produces a confidence score between 0 and 1.

The set of features is small and is not specific to any particular relation. This allows to train a model using a small amount of labeled data for one relation, and then use the model for scoring the candidates of all other relations. Since the supervised training stage needs to be run only once, it is a part of the system development, and the complete system remains unsupervised, as demonstrated in (Feldman and Rosenfeld 2006).

4 Entity Validation and Correction

In this paper we describe three different methods of validation and correction of relation arguments in the extracted instances. Two of them are "classical" and are based, respectively, on the knowledge-engineering, and on the statistical supervised approaches to the named entity recognition problems. The third is our novel approach, based on redundancy and corpus statistics.

The methods are implemented as components for SRES, called Entity Validators, inserted between the Instance Extractor and the Classifier. The result of applying Entity Validator to a candidate instance is an (optionally) fixed instance, with validity values attached to all slots. There are three validity values: *valid*, *invalid*, and *uncertain*.

The Classifier uses the validity values by converting them into two additional binary features, which are then able to influence the confidence of extractions.

We shall now describe the three different validators in details.

4.1 Small Rule-based NER validator

This validator is a small Perl script that checks whether a character string conforms to a set of simple regular expression patterns, and whether it appears inside lists of known named entities. There are two sets of regular expression patterns – for *Person* and for *Company* entity types, and three large lists – for known personal names, known companies, and "other known named entities", currently including locations, universities, and government agencies.

The manually written regular expression represent simple regularities in the internal structure of the entity types. For example, the patterns for *Person* include:

```
Person = KnownFirstName  [Initial]  LastName
Person = Honorific [FirstName] [Initial] LastName
Honorific = ("Mr" | "Ms" | "Dr" |…) ["."]
Initial = CapitalLetter ["."]
KnownFirstName = member of
                        KnownPersonalNamesList
FirstName = CapitalizedWord
LastName = CapitalizedWord
LastName = CapitalizedWord ["–"CapitalizedWord]
LastName = ("o" | "de" | …) "`"CapitalizedWord
        …
```

while the patterns for *Company* include:

```
Company = KnownCompanyName
Company = CompanyName CompanyDesignator
Company = CompanyName FrequentCompanySfx
KnownCompanyName = member of
                              KnownCompaniesList
CompanyName = CapitalizedWord +
CompanyDesignator = "inc" | "corp" | "co" | …
FrequentCompanySfx = "systems" | "software" | …
        …
```

The validator works in the following way: it receives a sentence with a labeled candidate entity of a specified entity type (which can be either *Person* or *Company*). It then applies all of the regular expression patterns to the labeled text and to its en-

closing context. It also checks for membership in the lists of known entities. If a boundary is incorrectly placed according to the patterns or to the lists, it is fixed. Then, the following result is returned:

Valid, if some pattern/list of the right entity type matched the candidate entity, while there were no matches for patterns/lists of other entity types.

Invalid, if no pattern/list of the right entity type matched the candidate entity, while there were matches for patterns/lists of other entity types.

Uncertain, otherwise, that is either if there were no matches at all, or if both correct and incorrect entity types matched.

The number of patterns is relatively small, and the whole component consists of about 300 lines in Perl and costs several person-days of knowledge engineering work. Despite its simplicity, we will show in the experimental section that it is quite effective, and even often outperforms the CRF-based NER component, described below.

4.2 CRF-based NER validator

This validator is built using a feature-rich CRF-based sequence classifier, trained upon an English dataset of the CoNLL 2003 shared task (Rosenfeld, Fresko et al. 2005). For the gazetteer lists it uses the same large lists as the rule-based component described above.

The validator receives a sentence with a labeled candidate entity of a specified entity type (which can be either *Person* or *Company*). It then sends the sentence to the CRF-based classifier, which labels all named entities it knows – *Dates*, *Times*, *Percents*, *Persons*, *Organizations*, and *Locations*. If the CRF classifier places the entity boundaries differently, they are fixed. Then, the following result is returned:

Valid, if CRF classification of the entity accords with the expected argument type.

Invalid, if CRF classification of the entity is different from the expected argument type.

Uncertain, otherwise, that is if the CRF classifier didn't recognize the entity at all.

4.3 Corpus-based NER validator

The goal of building the corpus-based NER validator is to provide the same level of performance as the supervised NER components, while requiring neither additional human supervision nor additional labeled corpora or other resources. There are several important facts that help achieve this goal.

First, the relation instances that are used as seeds for the pattern learning are known to contain correct instances of the right entity type. These instances can be used as seeds in their own right, for learning the patterns of occurrence of the corresponding entity types. Second, the entities in which we are interested usually appear in the corpus with a sufficient frequency. The validation is based on the first observation, while the boundary fixing on the second.

Corpus-based entity validation

There is a preparation stage, during which the information required for validation is extracted from the corpus. This information is the lists of all entities of every type that appears in the target relations. In order to extract these lists we use a simplified SRES. The entities are considered to be unary relations, and the seeds for them are taken from the slots of the target binary relations seeds. We don't use the Classifier on the extracted entity instances. Instead, for every extracted instance we record the number of different sentences the entity was extracted from.

During the validation process, the validator's task is to evaluate a given candidate entity instance. The validator compares the number of times the instance was extracted (during the preparation stage) by the patterns for the correct entity type, and by the patterns for all other entity types. The validator then returns

Valid, if the number of times the entity was extracted for the specified entity type is at least 5, and at least two times bigger than the number of times it was extracted for all other entity types.

Invalid, if the number of times the instance was extracted for the specified entity type is less than 5, and at least 2 times smaller than the number of times it was extracted for all other entity types.

Uncertain, otherwise, that is if it was never extracted at all, or extracted with similar frequency for both correct and wrong entity types.

Corpus-based correction of entity boundaries

Our entity boundaries correction mechanism is similar to the known statistical term extraction techniques (Su, Wu et al. 1994). It is based on the assumption that the component words of a term (an entity in our case) are more tightly bound to each other than to the context. In the statistical sense, this fact is expressed by a high mutual information between the adjacent words belonging to the same term.

There are two possible boundary fixes: removing words from the candidate entity, or adding words from the context to the entity. There is a significant practical difference between the two cases.

Assume that an entity boundary was placed too broadly, and included extra words. If this was a chance occurrence (and only such cases can be found by statistical methods), then the resulting sequence of tokens will be very infrequent, while its parts will have relatively high frequency. For example, consider a sequence "*Formerly Microsoft Corp*.", which is produced by mistakenly labeling "*Formerly*" as a proper noun by the PoS tagger. While it is easy to know from the frequencies that a boundary mistake was made, it is unclear (to the system) which part is the correct entity. But since the entity (one of the parts of the candidate) has a high frequency, there is a chance that the relation instance, in which the entity appears, will be repeated elsewhere in the corpus and will be extracted correctly there. Therefore, in such case, the simplest recourse is to simply label the entity as *Invalid*, and not to try fixing the boundaries.

On the other hand, if a word was missed from an entity (e.g., "*Beverly O*", instead of "*Beverly O ' Neill*"), the resulting sequence will be frequent. Moreover, it is quite probable that the same boundary mistake is made in many places, because the same sequence of tokens is being analyzed in all those places. Therefore, it makes sense to try to fix the bounary in this case, especially since it can be done simply and reliably: a word (or several words) is attached to the entity string if both their frequencies and their mutual information are above a threshold.

5 Experimental Evaluation

The experiments described in this paper aim to confirm the effectiveness of the proposed corpus-based relation argument validation and correction method, and to compare its performance with the classical knowledge-engineering-based and supervised-training-based methods. The experiments were performed with five relations:

Acquisition(BuyerCompany, AcquiredCompany),
Merger(Company1, Company2),
CEO_Of(Company, Person),
MayorOf(City, Person),
InventorOf(Person, Invention).

The data for the experiments were collected by the KnowItAll crawler. The data for the Acquisition and Merger consist of about 900,000 sentences for each of the two relations. The data for the bound relations consist of sentences, such that each contains one of a hundred values of the first (bound) attribute. Half of the hundred are frequent entities (>100,000 search engine hits), and another half are rare (<10,000 hits).

For evaluating the validators we randomly selected a set of 10000 sentences from the corpora for each of the relations, and manually evaluated the SRES results generated from these sentences. Four sets of results were evaluated: the baseline results produced without any NER validator, and three sets of results produced using three different NER validators. For the *InventorOf* relation, only the corpus-based validator results can be produced, since the other two NER components cannot be adapted to validate/correct entities of type *Invention*.

The results for the five relations are shown in the Figure 1. Several conclusions can be drawn from the graphs. First, all of the NER validators improve over the baseline SRES, sometimes as much as doubling the recall at the same level of precision. In most cases the three validators show roughly similar levels of performance. A notable difference is the *CEO_Of* relation, where the simple rule-based component performs much better than CRF, which performs yet better than the corpus-based component. The *CEO_Of* relation is tested as bound, which means that only the second relation argument, of type *Person*, is validated. The *Person* entities have much more rigid internal structure than the other entities – *Companies* and *Inventions*. Consequently, the best performing of

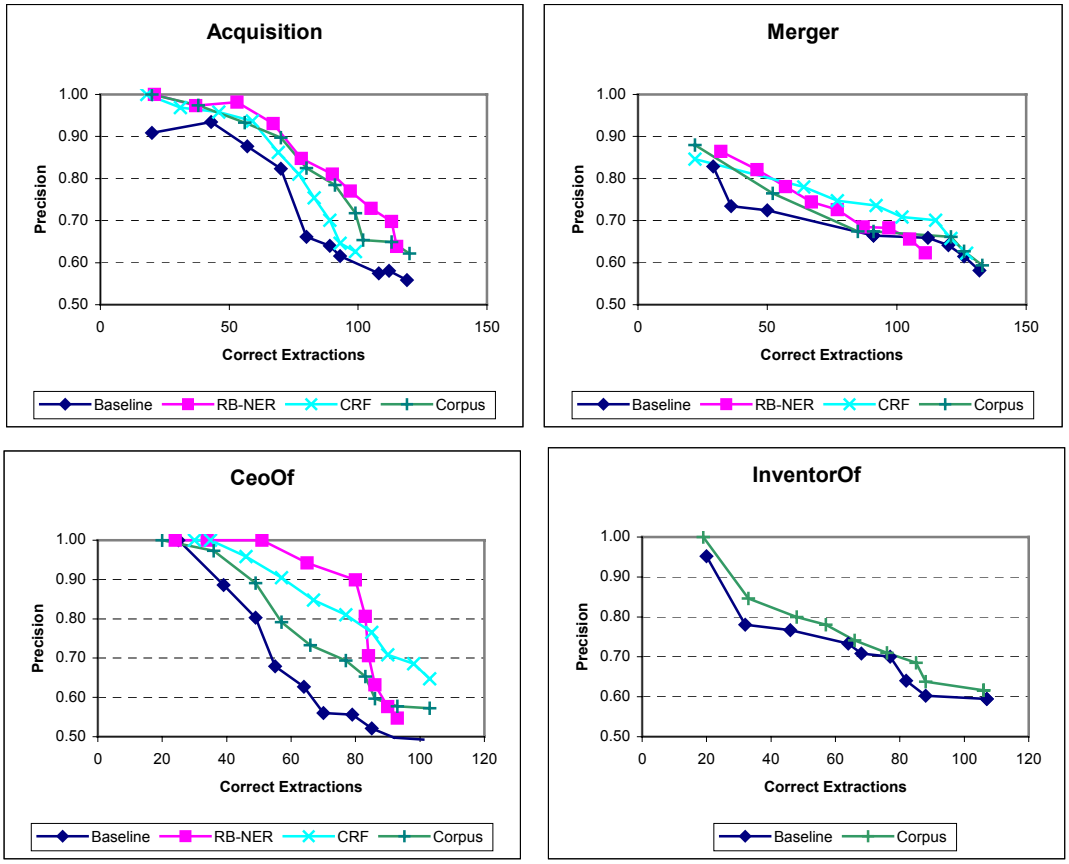


Figure 1. Comparison between Baseline-SRES and its extensions with three different NER validators: a simple Rule-Based one, a CRF-based statistical one, and a Corpus-based one.

the three validators is the rule-based, which directly tests this internal structure. The CRF-based validator is also able to take advantage of the structure, although in a weaker manner. The Corpus-based validator, however, works purely on the basis of context, entirely disregarding the internal structure of entities, and thus performs worst of all in this case. On the other hand, the Corpus-based validator is able to improve the results for the *Inventor* relation, which the other two validators are completely unable to do.

It is also of interest to compare the performance of CRF-based and the rule-based NER components in other cases. As can be seen, in most cases the rule-based component, despite its simplicity, outperforms the CRF-based one. The possible reason for this is that relation extraction setting is significantly different from the classical named entity recognition setting. A classical NER system is set to maximize the F_1 measure of all mentions of all entities in the corpus. A relation argument extractor, on the other hand, should maximize its performance on relation arguments, and apparently their statistical properties are often significantly different.

6 Conclusions

We have presented a novel method for validation and correction of relation arguments for the state-of-the-art unsupervised Web relation extraction system SRES. The method is based on corpus statistics and requires no human supervision and no additional corpus resources beyond the corpus that is used for relation extraction.

We showed experimentally the effectiveness of our method, which performed comparably to both simple rule-based NER and a statistical CRF-based NER in the task of validating *Companies*, and somewhat worse in the task of validating *Persons*,

due to its complete disregard of internal structure of entities. The ways to learn and use this structure in an unsupervised way are left for future research.

Our method also successfully validated the *Invention* entities, which are inaccessible to the other methods due to the lack of training data.

In our experiments we made use of a unique feature of SRES system – a feature-rich classifier that assigns confidence score to the candidate instances, basing its decisions on various features of the patterns and of the contexts from which the candidates were extracted. This architecture allows easy integration of the entity validation components as additional feature generators. We believe, however, that our results have greater applicability, and that the corpus statistics-based components can be added to RE systems with other architectures as well.

References

Agichtein, E. and L. Gravano (2000). *Snowball: Extracting Relations from Large Plain-Text Collections*. Proceedings of the 5th ACM International Conference on Digital Libraries (DL).

Brin, S. (1998). *Extracting Patterns and Relations from the World Wide Web*. WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT'98, Valencia, Spain.

Chen, J., D. Ji, C. L. Tan and Z. Niu (2005). *Unsupervised Feature Selection for Relation Extraction*. IJCNLP-05, Jeju Island, Korea.

Downey, D., M. Broadhead and O. Etzioni (2007). *Locating Complex Named Entities in Web Text*. IJCAI-07.

Etzioni, O., 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.

Feldman, R. and B. Rosenfeld (2006). *Boosting Unsupervised Relation Extraction by Using NER*. EMNLP-06, Sydney, Australia.

Feldman, R. and B. Rosenfeld (2006). *Self-Supervised Relation Extraction from the Web*. ISMIS-2006, Bari, Italy.

Hasegawa, T., S. Sekine and R. Grishman (2004). *Discovering Relations among Named Entities from Large Corpora*. ACL 2004.

Ravichandran, D. and E. Hovy (2002). *Learning Surface Text Patterns for a Question Answering System*. 40th ACL Conference.

Riloff, E. and R. Jones (1999). *Learning Dictionaries for Information Extraction by Multi-level Bootstrapping*. AAAI-99.

Rosenfeld, B., M. Fresko and R. Feldman (2005). *A Systematic Comparison of Feature-Rich Probabilistic Classifiers for NER Tasks*. PKDD.

Su, K.-Y., M.-W. Wu and J.-S. Chang (1994). A Corpus-based Approach to Automatic Compound Extraction. *Meeting of the Association for Computational Linguistics*: 242-247.

Beyond Projectivity: Multilingual Evaluation of Constraints and Measures on Non-Projective Structures

Jiří Havelka
Institute of Formal and Applied Linguistics
Charles University in Prague
Czech Republic
havelka@ufal.mff.cuni.cz

Abstract

Dependency analysis of natural language has gained importance for its applicability to NLP tasks. Non-projective structures are common in dependency analysis, therefore we need fine-grained means of describing them, especially for the purposes of machine-learning oriented approaches like parsing. We present an evaluation on twelve languages which explores several constraints and measures on non-projective structures. We pursue an edge-based approach concentrating on properties of individual edges as opposed to properties of whole trees. In our evaluation, we include previously unreported measures taking into account levels of nodes in dependency trees. Our empirical results corroborate theoretical results and show that an edge-based approach using levels of nodes provides an accurate and at the same time expressive means for capturing non-projective structures in natural language.

1 Introduction

Dependency analysis of natural language has been gaining an ever increasing interest thanks to its applicability in many tasks of NLP—a recent example is the dependency parsing work of McDonald et al. (2005), which introduces an approach based on the search for maximum spanning trees, capable of handling non-projective structures naturally.

The study of dependency structures occurring in natural language can be approached from two sides: by trying to delimit permissible dependency structures through formal constraints (for a recent review paper, see Kuhlmann and Nivre (2006)), or by providing their linguistic description (see e.g. Veselá et al. (2004) and Hajičová et al. (2004) for a linguistic analysis of non-projective constructions in Czech.[1])

We think that it is worth bearing in mind that neither syntactic structures in dependency treebanks, nor structures arising in machine-learning approaches, such as MST dependency parsing, need a priori fall into any formal subclass of dependency trees. We should therefore aim at formal means capable of describing all non-projective structures that are both expressive and fine-grained enough to be useful in statistical approaches, and at the same time suitable for an adequate linguistic description.[2]

Holan et al. (1998) first defined an infinite hierarchy of classes of dependency trees, going from projective to unrestricted dependency trees, based on the notion of gap degree for subtrees (cf. Section 3). Holan et al. (2000) present linguistic considerations concerning Czech and English with respect to this hierarchy (cf. also Section 6).

In this paper, we consider all constraints and measures evaluated by Kuhlmann and Nivre (2006)—with some minor variations, cf. Section 4.2. Ad-

[1] These two papers contain an error concerning an alternative condition of projectivity, which is rectified in Havelka (2005).

[2] The importance of such means becomes more evident from the asymptotically negligible proportion of projective trees to all dependency trees; there are super-exponentially many unrestricted trees compared to exponentially many projective trees on n nodes. Unrestricted dependency trees (i.e. labelled rooted trees) and projective dependency trees are counted by sequences A000169 and A006013 (offset 1), respectively, in the On-Line Encyclopedia of Sequences (Sloane, 2007).

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 608–615,
Prague, Czech Republic, June 2007.

ditionally, we introduce several measures not considered in their work. We also extend the empirical basis from Czech and Danish to twelve languages, which were made available in the CoNLL-X shared task on dependency parsing.

In our evaluation, we do not address the issue of what possible effects the annotations and/or conversions used when creating the data might have on non-projective structures in the different languages.

The newly considered measures have the first or both of the following desiderata: they are based on properties of individual non-projective edges (cf. Definition 3); and they take into account levels of nodes in dependency trees explicitly. None of the constraints and measures in Kuhlmann and Nivre (2006) take into account levels of nodes explicitly.

Level types of non-projective edges, introduced by Havelka (2005), have both desiderata. They provide an edge-based means of characterizing all non-projective structures; they also have some further interesting formal properties.

We propose a novel, more detailed measure, *level signatures* of non-projective edges, combining levels of nodes with the partitioning of gaps of non-projective edges into components. We derive a formal property of these signatures that links them to the constraint of well-nestedness, which is an extension of the result for level types (see also Havelka (2007b)).

The paper is organized as follows: Section 2 contains formal preliminaries; in Section 3 we review the constraint of projectivity and define related notions necessary in Section 4, where we define and discuss all evaluated constraints and measures; Section 5 describes our data and experimental setup; empirical results are presented in Section 6.

2 Formal preliminaries

Here we provide basic definitions and notation used in subsequent sections.

Definition 1 A *dependency tree* is a triple $(V, \rightarrow, \preceq)$, where V is a finite set of nodes, $\rightarrow$ a *dependency* relation on V, and $\preceq$ a total order on V.[3]

Relation $\rightarrow$ models linguistic dependency, and so represents a directed, rooted tree on V. There are many ways of characterizing rooted trees, we give here a characterization via the properties of $\rightarrow$: there is a *root* $r \in V$ such that $r \rightarrow^* v$ for all $v \in V$ and there is a unique *edge* $p \rightarrow v$ for all $v \in V$, $v \neq r$, and no edge into r. Relation $\rightarrow^*$ is the reflexive transitive closure of $\rightarrow$ and is usually called *subordination*.

For each node i we define its *level* as the length of the path $r \rightarrow^* i$; we denote it level_i. The symmetrization $\leftrightarrow = \rightarrow \cup \rightarrow^{-1}$ makes it possible to talk about edges (pairs of nodes i, j such that $i \rightarrow j$) without explicitly specifying the *parent* (*head*; i here) and the *child* (*dependent*; j here); so $\rightarrow$ represents directed edges and $\leftrightarrow$ undirected edges. To retain the ability to talk about the direction of edges, we define $\mathsf{Parent}_{i \leftrightarrow j} = \begin{cases} i & \text{if } i \rightarrow j \\ j & \text{if } j \rightarrow i \end{cases}$ and $\mathsf{Child}_{i \leftrightarrow j} = \begin{cases} j & \text{if } i \rightarrow j \\ i & \text{if } j \rightarrow i \end{cases}$. To make the exposition clearer by avoiding overuse of the symbol $\rightarrow$, we introduce notation for rooted *subtrees* not only for nodes, but also for edges: $\mathsf{Subtree}_i = \{v \in V \mid i \rightarrow^* v\}$, $\mathsf{Subtree}_{i \leftrightarrow j} = \{v \in V \mid \mathsf{Parent}_{i \leftrightarrow j} \rightarrow^* v\}$ (note that the subtree of an edge is defined relative to its parent node). To be able to talk concisely about the total order on nodes $\preceq$, we define *open intervals* whose endpoints need not be in a prescribed order $(i, j) = \{v \in V \mid \min_{\preceq}\{i, j\} \prec v \prec \max_{\preceq}\{i, j\}\}$.

3 Condition of projectivity

Projectivity of a dependency tree can be characterized both through the properties of its subtrees and through the properties of its edges.[4]

Definition 2 A dependency tree $T = (V, \rightarrow, \preceq)$ is *projective* if it satisfies the following equivalent conditions:

(Harper & Hays)
$i \rightarrow j \;\&\; v \in (i, j) \Longrightarrow v \in \mathsf{Subtree}_i$,

(Lecerf & Ihm)
$j \in \mathsf{Subtree}_i \;\&\; v \in (i, j) \Longrightarrow v \in \mathsf{Subtree}_i$,

(Fitialov)
$j_1, j_2 \in \mathsf{Subtree}_i \;\&\; v \in (j_1, j_2) \Longrightarrow v \in \mathsf{Subtree}_i$.

Otherwise T is *non-projective*.

[3] We adopt the following convention: nodes are drawn top-down according to their increasing level, with nodes on the same level being the same distance from the root; nodes are drawn from left to right according to the total order on nodes; edges are drawn as solid lines, paths as dotted curves.

[4] There are many other equivalent characterizations of projectivity, we give only three historically prominent ones.

It was Marcus (1965) who proved the equivalence of the conditions in Definition 2, proposed in the early 1960's (we denote them by the names of those to whom Marcus attributes their authorship).

We see that the antecedents of the projectivity conditions move from edge-focused to subtree-focused (i.e. from talking about dependency to talking about subordination).

It is the condition of Fitialov that has been mostly explored when studying so-called relaxations of projectivity. (The condition is usually worded as follows: A dependency tree is projective if the nodes of all its subtrees constitute contiguous intervals in the total order on nodes.)

However, we find the condition of Harper & Hays to be the most appealing from the linguistic point of view because it gives prominence to the primary notion of dependency edges over the derived notion of subordination. We therefore use an edge-based approach whenever we find it suitable.

To that end, we need the notion of a non-projective edge and its gap.

Definition 3 For any edge $i \leftrightarrow j$ in a dependency tree T we define its *gap* as follows

$$\mathsf{Gap}_{i \leftrightarrow j} = \{v \in V \mid v \in (i,j) \ \&\ v \notin \mathsf{Subtree}_{i \leftrightarrow j}\} \ .$$

An edge with an empty gap is *projective*, an edge whose gap is non-empty is *non-projective*.[5]

We see that non-projective are those edges $i \leftrightarrow j$ for which there is a node v such that together they violate the condition of Harper & Hays; we group all such nodes v into $\mathsf{Gap}_{i \leftrightarrow j}$, the gap of the non-projective edge $i \leftrightarrow j$.

The notion of gap is defined differently for subtrees of a dependency tree (Holan et al., 1998; Bodirsky et al., 2005). There it is defined through the nodes of the whole dependency tree not in the considered subtree that intervene between its nodes in the total order on nodes $\preceq$.

4 Relaxations of projectivity: evaluated constraints and measures

In this section we present all constraints and measures on dependency trees that we evaluate empirically in Section 6. First we give definitions of global constraints on dependency trees, then we present measures of non-projectivity based on properties of individual non-projective edges (some of the edge-based measures have corresponding tree-based counterparts, however we do not discuss them in detail).

4.1 Tree constraints

We consider the following three global constraints on dependency trees: projectivity, planarity, and well-nestedness. All three constraints can be applied to more general structures, e.g. dependency forests or even general directed graphs. Here we adhere to their primary application to dependency trees.

Definition 4 A dependency tree T is *non-planar* if there are two edges $i_1 \leftrightarrow j_1$, $i_2 \leftrightarrow j_2$ in T such that

$$i_1 \in (i_2, j_2) \ \ \&\ \ i_2 \in (i_1, j_1) \ .$$

Otherwise T is *planar*.

Planarity is a relaxation of projectivity that corresponds to the "no crossing edges" constraint. Although it might get confused with projectivity, it is in fact a strictly weaker constraint. Planarity is equivalent to projectivity for dependency trees with their root node at either the left or right fringe of the tree.

Planarity is a recent name for a constraint studied under different names already in the 1960's—we are aware of independent work in the USSR (*weakly non-projective trees*; see the survey paper by Dikovsky and Modina (2000) for references) and in Czechoslovakia (*smooth trees*; Nebeský (1979) presents a survey of his results).

Definition 5 A dependency tree T is *ill-nested* if there are two non-projective edges $i_1 \leftrightarrow j_1$, $i_2 \leftrightarrow j_2$ in T such that

$$i_1 \in \mathsf{Gap}_{i_2 \leftrightarrow j_2} \ \ \&\ \ i_2 \in \mathsf{Gap}_{i_1 \leftrightarrow j_1} \ .$$

Otherwise T is *well-nested*.

Well-nestedness was proposed by Bodirsky et al. (2005). The original formulation forbids interleaving of disjoint subtrees in the total order on nodes; we present an equivalent formulation in terms of non-projective edges, derived in (Havelka, 2007b).

Figure 1 illustrates the subset hierarchy between classes of dependency trees satisfying the particular constraints:

projective $\subsetneq$ planar $\subsetneq$ well-nested $\subsetneq$ unrestricted

[5] In figures with sample configurations we adopt this convention: for a non-projective edge, we draw all nodes in its gap explicitly and assume that no node on any path crossing the span of the edge lies in the interval delimited by its endpoints.

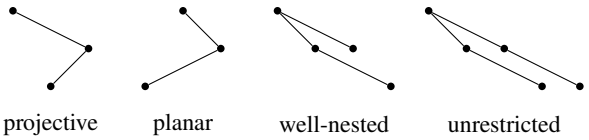


Figure 1: Sample dependency trees (trees satisfy corresponding constraints and violate all preceding ones)

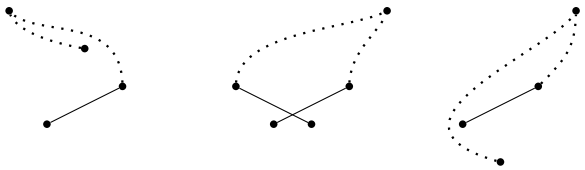


Figure 2: Sample configurations with non-projective edges of different level types

4.2 Edge measures

The first two measures are based on two ways of partitioning the gap of a non-projective edge—into intervals and into components. The third measure, level type, is based on levels of nodes. We also propose a novel measure combining levels of nodes and the partitioning of gaps into components.

Definition 6 For any edge $i \leftrightarrow j$ in a dependency tree T we define its *interval degree* as follows

$$\mathsf{ideg}_{i\leftrightarrow j} = \text{number of intervals in } \mathsf{Gap}_{i\leftrightarrow j} \ .$$

By an interval we mean a contiguous interval in $\preceq$, i.e. a maximal set of nodes comprising all nodes between its endpoints in the total order on nodes $\preceq$.

This measure corresponds to the tree-based *gap degree* measure in (Kuhlmann and Nivre, 2006), which was first introduced in (Holan et al., 1998)—there it is defined as the maximum over gap degrees of all subtrees of a dependency tree (the gap degree of a subtree is the number of contiguous intervals in the gap of the subtree). The interval degree of an edge is bounded from above by the gap degree of the subtree rooted in its parent node.

Definition 7 For any edge $i \leftrightarrow j$ in a dependency tree T we define its *component degree* as follows

$$\mathsf{cdeg}_{i\leftrightarrow j} = \text{number of components in } \mathsf{Gap}_{i\leftrightarrow j} \ .$$

By a component we mean a connected component in the relation $\leftrightarrow$, in other words a weak component in the relation $\rightarrow$ (we consider relations induced on the set $\mathsf{Gap}_{i\leftrightarrow j}$ by relations on T).

This measure was introduced by Nivre (2006); Kuhlmann and Nivre (2006) call it *edge degree*. Again, they define it as the maximum over all edges.

Each component of a gap can be represented by a single node, its *root* in the dependency relation induced on the nodes of the gap (i.e. a node of the component closest to the root of the whole tree). Note that a component need not constitute a full subtree of the dependency tree (there may be nodes in the subtree of the component root that lie outside the span of the particular non-projective edge).

Definition 8 The *level type* (or just *type*) of a non-projective edge $i \leftrightarrow j$ in a dependency tree T is defined as follows

$$\mathsf{Type}_{i\leftrightarrow j} = \mathsf{level}_{\mathsf{Child}_{i\leftrightarrow j}} - \min_{n\in\mathsf{Gap}_{i\leftrightarrow j}} \mathsf{level}_n \ .$$

The level type of an edge is the relative distance in levels of its child node and a node in its gap closest to the root; there may be more than one node witnessing an edge's type. For sample configurations see Figure 2. Properties of level types are presented in Havelka (2005; 2007b).[6]

We propose a new measure combining level types and component degrees. (We do not use interval degrees, i.e. the partitioning of gaps into intervals, because we cannot specify a unique representative of an interval with respect to the tree structure.)

Definition 9 The *level signature* (or just *signature*) of an edge $i \leftrightarrow j$ in a dependency tree T is a mapping $\mathsf{Signature}_{i\leftrightarrow j} : \mathcal{P}(V) \rightarrow {}^{\mathbb{Z}}\mathbb{N}_0$ defined as follows

$$\mathsf{Signature}_{i\leftrightarrow j} = \{\mathsf{level}_{\mathsf{Child}_{i\leftrightarrow j}} - \mathsf{level}_r \mid r \text{ is component root in } \mathsf{Gap}_{i\leftrightarrow j}\} \ .$$

(The right-hand side is considered as a multiset, i.e. elements may repeat.) We call the elements of a signature *component levels*.

The signature of an edge is a multiset consisting of the relative distances in levels of all component roots in its gap from its child node.

Further, we disregard any possible orderings on signatures and concentrate only on the relative distances in levels. We present signatures as non-

[6] For example, presence of non-projective edges of nonnegative level type in equivalent to non-projectivity of a dependency tree; moreover, all such edges can be found in linear time.

decreasing sequences and write them in angle brackets $\langle\ \rangle$, component levels separated by commas (by doing so, we avoid combinatorial explosion).

Notice that level signatures subsume level types: the level type of a non-projective edge is the component level of any of possibly several component roots closest to the root of the whole tree. In other words, the level type of an edge is equal to the largest component level occurring in its level signature.

Level signatures share interesting formal properties with level types of non-projective edges. The following result is a direct extension of the results presented in Havelka (2005; 2007b).

Theorem 10 *Let $i \leftrightarrow j$ be a non-projective edge in a dependency tree T. For any component c in $\mathsf{Gap}_{i\leftrightarrow j}$ represented by root r_c with component level $l_c \leq 0$ (< 0) there is a non-projective edge $v \rightarrow r_c$ in T with $\mathsf{Type}_{v\leftrightarrow r_c} \geq 0$ (> 0) such that either $i \in \mathsf{Gap}_{v\leftrightarrow r_c}$, or $j \in \mathsf{Gap}_{v\leftrightarrow r_c}$.*

PROOF. From the assumptions $l_c \leq 0$ and $r_c \in \mathsf{Gap}_{i\leftrightarrow j}$ the parent v of node r_c lies outside the span of the edge $i \leftrightarrow j$, hence $v \notin \mathsf{Gap}_{i\leftrightarrow j}$. Thus either $i \in (v, r_c)$, or $j \in (v, r_c)$. Since $\mathsf{level}_v \geq \mathsf{level}_{\mathsf{Parent}_{i\leftrightarrow j}}$, we have that $\mathsf{Parent}_{i\leftrightarrow j} \notin \mathsf{Subtree}_v$, and so either $i \in \mathsf{Gap}_{v\leftrightarrow r_c}$, or $j \in \mathsf{Gap}_{v\leftrightarrow r_c}$. Finally from $l_c = \mathsf{level}_{\mathsf{Child}_{i\leftrightarrow j}} - \mathsf{level}_{r_c} \leq 0$ (< 0) we get $\mathsf{level}_{r_c} - \mathsf{level}_{\mathsf{Child}_{i\leftrightarrow j}} \geq 0$ (> 0), hence $\mathsf{Type}_{v\leftrightarrow r_c} \geq 0$ (> 0). ∎

This result links level signatures to well-nestedness: it tells us that whenever an edge's signature contains a nonpositive component level, the whole dependency tree is ill-nested (because then there are two edges satisfying Definition 5).

All discussed edge measures take integer values: interval and component degrees take only nonnegative values, level types and level signatures take integer values (in all cases, their absolute values are bounded by the size of the whole dependency tree). Both interval and component degrees are defined also for projective edges (for which they take value 0), level type is undefined for projective edges, however the level signature of projective edges is defined—it is the empty multiset/sequence.

5 Data and experimental setup

We evaluate all constraints and measures described in the previous section on 12 languages, whose treebanks were made available in the CoNLL-X shared task on dependency parsing (Buchholz and Marsi, 2006). In alphabetical order they are: Arabic, Bulgarian, Czech, Danish, Dutch, German, Japanese, Portuguese, Slovene, Spanish, Swedish, and Turkish (Hajič et al., 2004; Simov et al., 2005; Böhmová et al., 2003; Kromann, 2003; van der Beek et al., 2002; Brants et al., 2002; Kawata and Bartels, 2000; Afonso et al., 2002; Džeroski et al., 2006; Civit Torruella and Martí Antonín, 2002; Nilsson et al., 2005; Oflazer et al., 2003).[7] We do not include Chinese, which is also available in this data format, because all trees in this data set are projective.

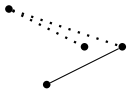


Figure 3: Sample non-projective tree considered planar in empirical evaluation

We take the data "as is", although we are aware that structures occurring in different languages depend on the annotations and/or conversions used (some languages were not originally annotated with dependency syntax, but only converted to a unified dependency format from other representations).

The CoNLL data format is a simple tabular format for capturing dependency analyses of natural language sentences. For each sentence, it uses a technical root node to which dependency analyses of parts of the sentence (possibly several) are attached. Equivalently, the representation of a sentence can be viewed as a forest consisting of dependency trees.

By conjoining partial dependency analyses under one technical root node, we let all their edges interact. Since the technical root comes before the sentence itself, no new non-projective edges are introduced. However, edges from technical roots may introduce non-planarity. Therefore, in our empirical evaluation we disregard all such edges when counting trees conforming to the planarity constraint; we also exclude them from the total numbers of edges. Figure 3 exemplifies how this may affect counts of non-planar trees;[8] cf. also the remark after Definition 4. Counts of well-nested trees are not affected.

[7] All data sets are the train parts of the CoNLL-X shared task.

[8] The sample tree is non-planar according to Definition 4, however we do not consider it as such, because all pairs of "crossing edges" involve an edge from the technical root (edges from the technical root are depicted as dotted lines).

6 Empirical results

Our complete results for global constraints on dependency trees are given in Table 1. They confirm the findings of Kuhlmann and Nivre (2006): planarity seems to be almost as restrictive as projectivity; well-nestedness, on the other hand, covers large proportions of trees in all languages.

In contrast to global constraints, properties of individual non-projective edges allow us to pinpoint the causes of non-projectivity. Therefore they provide a means for a much more fine-grained classification of non-projective structures occurring in natural language. Table 2 presents highlights of our analysis of edge measures.

Both interval and component degrees take generally low values. On the other hand, Holan et al. (1998; 2000) show that at least for Czech neither of these two measures can in principle be bounded.

Taking levels of nodes into account seems to bring both better accuracy and expressivity. Since level signatures subsume level types as their last components, we only provide counts of edges of positive, nonpositive, and negative level types. For lack of space, we do not present full distributions of level types nor of level signatures.

Positive level types give an even better fit with real linguistic data than the global constraint of well-nestedness (an ill-nested tree need not contain a non-projective edge of nonpositive level type; cf. Theorem 10). For example, in German less than one tenth of ill-nested trees contain an edge of nonpositive level type. Minimum negative level types for Czech, Slovene, Swedish, and Turkish are respectively -1, -5, -2, and -4.

Level signatures combine level types and component degrees, and so give an even more detailed picture of the gaps of non-projective edges. In some languages the actually occurring signatures are quite limited, in others there is a large variation.

Because we consider it linguistically relevant, we also count how many non-projective edges contain in their gaps a component rooted in an ancestor of the edge (an *ancestor* of an edge is any node on the path from the root of the whole tree to the parent node of the edge). The proportions of such non-projective edges vary widely among languages and for some this property seems highly important.

Empirical evidence shows that edge measures of non-projectivity taking into account levels of nodes fit very well with linguistic data. This supports our theoretical results and confirms that properties of non-projective edges provide a more accurate as well as expressive means for describing non-projective structures in natural language than the constraints and measures considered by Kuhlmann and Nivre (2006).

7 Conclusion

In this paper, we evaluate several constraints and measures on non-projective dependency structures. We pursue an edge-based approach giving prominence to properties of individual edges. At the same time, we consider levels of nodes in dependency trees. We find an edge-based approach also more appealing linguistically than traditional approaches based on properties of whole dependency trees or their subtrees. Furthermore, edge-based properties allow machine-learning techniques to model global phenomena locally, resulting in less sparse models.

We propose a new edge measure of non-projectivity, level signatures of non-projective edges. We prove that, analogously to level types, they relate to the constraint of well-nestedness.

Our empirical results on twelve languages can be summarized as follows: Among the global constraints, well-nestedness fits best with linguistic data. Among edge measures, the previously unreported measures taking into account levels of nodes stand out. They provide both the best fit with linguistic data of all constraints and measures we have considered, as well as a substantially more detailed capability of describing non-projective structures.

The interested reader can find a more in-depth and broader-coverage discussion of properties of dependency trees and their application to natural language syntax in (Havelka, 2007a).

As future work, we plan to investigate more languages and carry out linguistic analyses of non-projective structures in some of them. We will also apply our results to statistical approaches to NLP tasks, such as dependency parsing.

Acknowledgement The research reported in this paper was supported by Project No. 1ET201120505 of the Ministry of Education of the Czech Republic.

Language	Arabic	Bulgarian	Czech	Danish	Dutch	German	Japanese	Portuguese	Slovene	Spanish	Swedish	Turkish
ill-nested	1		79	6	15	416		7	3		71	14
non-planar	150	677	13783	787	4115	10865	1	1713	283	56	1076	556
non-projective	163	690	16831	811	4865	10883	902	1718	340	57	1079	580
proportion of all (%)	11.16%	5.38%	23.15%	15.63%	36.44%	27.75%	5.29%	18.94%	22.16%	1.72%	9.77%	11.6%
all	1460	12823	72703	5190	13349	39216	17044	9071	1534	3306	11042	4997

Table 1: Counts of dependency trees violating global constraints of well-nestedness, planarity, and projectivity; the last line gives the total numbers of dependency trees. (An empty cell means count zero.)

Language	Arabic	Bulgarian	Czech	Danish	Dutch	German	Japanese	Portuguese	Slovene	Spanish	Swedish	Turkish
ideg $= 1$	211	724	23376	940	10209	14605	1570	2398	548	58	1829	813
ideg $= 2$		1	189	5	349	1198	81	272	2	1	46	27
ideg $= 3$			3		8	37	12	24			9	1
cdeg $= 1$	200	723	23190	842	10264	13107	1484	2466	531	59	1546	623
cdeg $= 2$	10	1	292	78	238	2206	143	151	11		204	146
cdeg $= 3$	1	1	66	22	47	434	26	64	2		76	55
Type > 0	211	725	23495	942	10564	15803	1667	2699	547	59	1847	833
Type ≤ 0			75	3	2	41		3	3		50	8
Type < 0			4						2		15	2
Signature / count	$\langle 1\rangle$ / 92	$\langle 2\rangle$ / 674	$\langle 2\rangle$ / 18507	$\langle 2\rangle$ / 555	$\langle 2\rangle$ / 8061	$\langle 2\rangle$ / 8407	$\langle 1\rangle$ / 466	$\langle 2\rangle$ / 1670	$\langle 2\rangle$ / 384	$\langle 2\rangle$ / 46	$\langle 2\rangle$ / 823	$\langle 2\rangle$ / 341
	$\langle 2\rangle$ / 56	$\langle 3\rangle$ / 32	$\langle 1\rangle$ / 2886	$\langle 1\rangle$ / 115	$\langle 3\rangle$ / 1461	$\langle 1\rangle$ / 3112	$\langle 2\rangle$ / 209	$\langle 1\rangle$ / 571	$\langle 1\rangle$ / 67	$\langle 3\rangle$ / 7	$\langle 1\rangle$ / 530	$\langle 1\rangle$ / 189
	$\langle 3\rangle$ / 18	$\langle 1\rangle$ / 10	$\langle 3\rangle$ / 1515	$\langle 3\rangle$ / 100	$\langle 1\rangle$ / 512	$\langle 1,1\rangle$ / 1503	$\langle 4\rangle$ / 186	$\langle 3\rangle$ / 208	$\langle 3\rangle$ / 45	$\langle 4\rangle$ / 4	$\langle 3\rangle$ / 114	$\langle 1,1\rangle$ / 91
	$\langle 4\rangle$ / 10	$\langle 4\rangle$ / 5	$\langle 4\rangle$ / 154	$\langle 1,1\rangle$ / 63	$\langle 4\rangle$ / 201	$\langle 3\rangle$ / 1397	$\langle 3\rangle$ / 183	$\langle 1,1\rangle$ / 113	$\langle 4\rangle$ / 13	$\langle 1\rangle$ / 2	$\langle 1,1\rangle$ / 94	$\langle 3\rangle$ / 53
	$\langle 1,1\rangle$ / 8	$\langle 5\rangle$ / 2	$\langle 1,1\rangle$ / 115	$\langle 4\rangle$ / 41	$\langle 1,1\rangle$ / 118	$\langle 2,2\rangle$ / 476	$\langle 5\rangle$ / 126	$\langle 1,1,1\rangle$ / 44	$\langle 5\rangle$ / 12		$\langle 0\rangle$ / 31	$\langle 2,2\rangle$ / 31
	$\langle 5\rangle$ / 7	$\langle 1,1,1\rangle$ / 1	$\langle 0\rangle$ / 70	$\langle 5\rangle$ / 16	$\langle 2,2\rangle$ / 52	$\langle 1,1,1\rangle$ / 312	$\langle 6\rangle$ / 113	$\langle 2,2\rangle$ / 29	$\langle 1,1\rangle$ / 6		$\langle 1,3\rangle$ / 27	$\langle 1,1,1\rangle$ / 29
	$\langle 6\rangle$ / 6	$\langle 1,1\rangle$ / 1	$\langle 2,2\rangle$ / 58	$\langle 1,1,1\rangle$ / 16	$\langle 1,1,1\rangle$ / 25	$\langle 4\rangle$ / 136	$\langle 7\rangle$ / 78	$\langle 2,2,2\rangle$ / 13	$\langle 6\rangle$ / 4		$\langle 1,1,1\rangle$ / 25	$\langle 4\rangle$ / 19
	$\langle 7\rangle$ / 4		$\langle 1,1,1\rangle$ / 48	$\langle 2,2\rangle$ / 7	$\langle 5\rangle$ / 23	$\langle 3,3\rangle$ / 98	$\langle 1,1\rangle$ / 63	$\langle 4\rangle$ / 12	$\langle 1,1,1,1\rangle$ / 4		$\langle 4\rangle$ / 21	$\langle 2,2,2\rangle$ / 10
	$\langle 2,2\rangle$ / 2		$\langle 2,4\rangle$ / 44	$\langle 6\rangle$ / 6	$\langle 1,3\rangle$ / 16	$\langle 2,2,2\rangle$ / 69	$\langle 8\rangle$ / 49	$\langle 1,1,1,1\rangle$ / 7	$\langle 7\rangle$ / 2		$\langle 1,2\rangle$ / 19	$\langle 3,3\rangle$ / 6
	$\langle 9\rangle$ / 1		$\langle 1,3\rangle$ / 32	$\langle 2,2,2\rangle$ / 6	$\langle 3,3\rangle$ / 15	$\langle 1,1,1,1\rangle$ / 59	$\langle 9\rangle$ / 35	$\langle 1,1,1,1,1\rangle$ / 6	$\langle 1,1,3\rangle$ / 2		$\langle 2,2\rangle$ / 16	$\langle 2,2,2,2\rangle$ / 6
	⋮		⋮	⋮	⋮	⋮	⋮	⋮	⋮		⋮	⋮
ancestor comp. root	39	711	20035	703	9781	10128	0	1832	392	57	950	345
only ancestor comp. r.	39	711	19913	685	9697	9526	0	1820	386	57	857	340
non-projective	211	725	23570	945	10566	15844	1667	2702	550	59	1897	841
proportion of all (%)	0.42%	0.41%	2.13%	1.06%	5.9%	2.4%	1.32%	1.37%	2.13%	0.07%	1.05%	1.61%
all	50097	177394	1105437	89171	179063	660394	126511	197607	25777	86028	180425	52273

Table 2: Counts for edge measures *interval degree*, *component degree* (for values from 1 to 3; larger values are not included), *level type* (for positive, nonpositive, and negative values), *level signature* (up to 10 most frequent values), and numbers of edges with *ancestor component roots* in their gaps and solely with ancestor component roots in their gaps; the second to last line gives the total numbers of non-projective edges, the last line gives the total numbers of all edges—we exclude edges from technical roots. (The listings need not be exhaustive; an empty cell means count zero.)

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*, volume 20 of *Text, Speech and Language Technology*. Kluwer Academic Publishers, Dordrecht.

S. Afonso, E. Bick, R. Haber, and D. Santos. 2002. "Floresta sintá(c)tica": a treebank for Portuguese. In *Proceedings of the 3rd Intern. Conf. on Language Resources and Evaluation (LREC)*, pages 1698–1703.

Manuel Bodirsky, Marco Kuhlmann, and Matthias Möhl. 2005. Well-nested drawings as models of syntactic structure. In *Proceedings of Tenth Conference on Formal Grammar and Ninth Meering on Mathematics of Language*.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (2003), chapter 7.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER treebank. In *Proceedings of the 1st Workshop on Treebanks and Linguistic Theories (TLT)*.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL-X*. SIGNLL.

M. Civit Torruella and M^{a} A. Martí Antonín. 2002. Design principles for a Spanish treebank. In *Proceedings of the 1st Workshop on Treebanks and Linguistic Theories (TLT)*.

Alexander Dikovsky and Larissa Modina. 2000. Dependencies on the other side of the Curtain. *Traitement Automatique des Langues (TAL)*, 41(1):67–96.

S. Džeroski, T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtsky, and A. Žele. 2006. Towards a Slovene dependency treebank. In *Proceedings of the 5th Intern. Conf. on Language Resources and Evaluation (LREC)*.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proceedings of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

Eva Hajičová, Jiří Havelka, Petr Sgall, Kateřina Veselá, and Daniel Zeman. 2004. Issues of Projectivity in the Prague Dependency Treebank. *Prague Bulletin of Mathematical Linguistics*, 81:5–22.

Jiří Havelka. 2005. Projectivity in Totally Ordered Rooted Trees: An Alternative Definition of Projectivity and Optimal Algorithms for Detecting Non-Projective Edges and Projectivizing Totally Ordered Rooted Trees. *Prague Bulletin of Mathematical Linguistics*, 84:13–30.

Jiří Havelka. 2007a. *Mathematical Properties of Dependency Trees and their Application to Natural Language Syntax*. Ph.D. thesis, Institute of Formal and Applied Linguistics, Charles University in Prague, Czech Republic.

Jiří Havelka. 2007b. Relationship between Non-Projective Edges, Their Level Types, and Well-Nestedness. In *Proceedings of HLT/NAACL; Companion Volume, Short Papers*, pages 61–64.

Tomáš Holan, Vladislav Kuboň, Karel Oliva, and Martin Plátek. 1998. Two Useful Measures of Word Order Complexity. In Alain Polguère and Sylvain Kahane, editors, *Proceedings of Dependency-Based Grammars Workshop, COLING/ACL*, pages 21–28.

Tomáš Holan, Vladislav Kuboň, Karel Oliva, and Martin Plátek. 2000. On Complexity of Word Order. *Traitement Automatique des Langues (TAL)*, 41(1):273–300.

Y. Kawata and J. Bartels. 2000. Stylebook for the Japanese treebank in VERBMOBIL. Verbmobil-Report 240, Seminar für Sprachwissenschaft, Universität Tübingen.

M. T. Kromann. 2003. The Danish dependency treebank and the underlying linguistic theory. In *Proceedings of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*.

Marco Kuhlmann and Joakim Nivre. 2006. Mildly Non-Projective Dependency Structures. In *Proceedings of COLING/ACL*, pages 507–514.

Solomon Marcus. 1965. Sur la notion de projectivité [On the notion of projectivity]. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 11:181–192.

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

Ladislav Nebeský. 1979. Graph theory and linguistics (chapter 12). In R. J. Wilson and L. W. Beineke, editors, *Applications of Graph Theory*, pages 357–380. Academic Press.

J. Nilsson, J. Hall, and J. Nivre. 2005. MAMBA meets TIGER: Reconstructing a Swedish treebank from antiquity. In *Proceedings of the NODALIDA Special Session on Treebanks*.

Joakim Nivre. 2006. Constraints on Non-Projective Dependency Parsing. In *Proceedings of EACL*, pages 73–80.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (2003), chapter 15.

K. Simov, P. Osenova, A. Simov, and M. Kouylekov. 2005. Design and implementation of the Bulgarian HPSG-based treebank. In *Journal of Research on Language and Computation – Special Issue*, pages 495–522. Kluwer Academic Publishers.

Neil J. A. Sloane. 2007. On-Line Encyclopedia of Integer Sequences. Published electronically at `www.research.att.com/~njas/sequences/`.

L. van der Beek, G. Bouma, R. Malouf, and G. van Noord. 2002. The Alpino dependency treebank. In *Computational Linguistics in the Netherlands (CLIN)*.

Kateřina Veselá, Jiří Havelka, and Eva Hajičová. 2004. Condition of Projectivity in the Underlying Dependency Structures. In *Proceedings of COLING*, pages 289–295.

Self-Training for Enhancement and Domain Adaptation of Statistical Parsers Trained on Small Datasets

Roi Reichart
ICNC
Hebrew University of Jerusalem
roiri@cs.huji.ac.il

Ari Rappoport
Institute of Computer Science
Hebrew University of Jerusalem
arir@cs.huji.ac.il

Abstract

Creating large amounts of annotated data to train statistical PCFG parsers is expensive, and the performance of such parsers declines when training and test data are taken from different domains. In this paper we use self-training in order to improve the quality of a parser and to adapt it to a different domain, using only small amounts of manually annotated seed data. We report significant improvement both when the seed and test data are in the same domain and in the out-of-domain adaptation scenario. In particular, we achieve 50% reduction in annotation cost for the in-domain case, yielding an improvement of 66% over previous work, and a 20-33% reduction for the domain adaptation case. This is the first time that self-training with small labeled datasets is applied successfully to these tasks. We were also able to formulate a characterization of when self-training is valuable.

1 Introduction

State of the art statistical parsers (Collins, 1999; Charniak, 2000; Koo and Collins, 2005; Charniak and Johnson, 2005) are trained on manually annotated treebanks that are highly expensive to create. Furthermore, the performance of these parsers decreases as the distance between the genres of their training and test data increases. Therefore, enhancing the performance of parsers when trained on *small* manually annotated datasets is of great importance, both when the seed and test data are taken from the same domain (the *in-domain* scenario) and when they are taken from different domains (the *out-of-domain* or *parser adaptation* scenario). Since the problem is the expense in manual annotation, we define 'small' to be 100-2,000 sentences, which are the sizes of sentence sets that can be manually annotated by constituent structure in a few hours[1].

Self-training is a method for using unannotated data when training supervised models. The model is first trained using manually annotated ('seed') data, then the model is used to automatically annotate a pool of unannotated ('self-training') data, and then the manually and automatically annotated datasets are combined to create the training data for the final model. Self-training of parsers trained on small datasets is of enormous potential practical importance, due to the huge amounts of unannotated data that are becoming available today and to the high cost of manual annotation.

In this paper we use self-training to enhance the performance of a generative statistical PCFG parser (Collins, 1999) for both the in-domain and the parser adaptation scenarios, using only small amounts of manually annotated data. We perform four experiments, examining all combinations of in-domain and out-of-domain seed and self-training data.

Our results show that self-training is of substantial benefit for the problem. In particular, we present:

- 50% reduction in annotation cost when the seed and test data are taken from the same domain, which is 66% higher than any previous result with small manually annotated datasets.

[1] We note in passing that quantitative research on the cost of annotation using various annotation schemes is clearly lacking.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 616–623,
Prague, Czech Republic, June 2007.

- The first time that self-training improves a generative parser when the seed and test data are from the same domain.

- 20-33% reduction in annotation cost when the seed and test data are from different domains.

- The first time that self-training succeeds in adapting a generative parser between domains using a small manually annotated dataset.

- The first formulation (related to the number of unknown words in a sentence) of when self-training is valuable.

Section 2 discusses previous work, and Section 3 compares in-depth our protocol to a previous one. Sections 4 and 5 present the experimental setup and our results, and Section 6 analyzes the results in an attempt to shed light on the phenomenon of self-training.

2 Related Work

Self-training might seem a strange idea: why should a parser trained on its own output learn anything new? Indeed, (Clark et al., 2003) applied self-training to POS-tagging with poor results, and (Charniak, 1997) applied it to a generative statistical PCFG parser trained on a large seed set (40K sentences), without any gain in performance.

Recently, (McClosky et al., 2006a; McClosky et al., 2006b) have successfully applied self-training to various parser adaptation scenarios using the reranking parser of (Charniak and Johnson, 2005). A reranking parser (see also (Koo and Collins, 2005)) is a layered model: the base layer is a generative statistical PCFG parser that creates a ranked list of k parses (say, 50), and the second layer is a reranker that reorders these parses using more detailed features. McClosky et al (2006a) use sections 2-21 of the WSJ PennTreebank as seed data and between 50K to 2,500K unlabeled NANC corpus sentences as self-training data. They train the PCFG parser and the reranker with the manually annotated WSJ data, and parse the NANC data with the 50-best PCFG parser. Then they proceed in two directions. In the first, they reorder the 50-best parse list with the reranker to create a new 1-best list. In the second, they leave the 1-best list produced by the generative PCFG parser untouched. Then they combine the 1-best list (each direction has its own list) with the WSJ training set, to retrain the PCFG parser. The final PCFG model and the reranker (trained only on annotated WSJ material) are then used to parse the test section (23) of WSJ.

There are two major differences between these papers and the current one, stemming from their usage of a reranker and of large seed data. First, when their 1-best list of the base PCFG parser was used as self training data for the PCFG parser (the second direction), the performance of the base parser did not improve. It had improved only when the 1-best list of the *reranker* was used. In this paper we show how the 1-best list of a base (generative) PCFG parser can be used as a self-training material for the base parser itself and enhance its performance, without using any reranker. This reveals a noteworthy characteristic of generative PCFG models and offers a potential direction for parser improvement, since the quality of a parser-reranker combination critically depends on that of the base parser.

Second, these papers did not explore self-training when the seed is small, a scenario whose importance has been discussed above. In general, PCFG models trained on small datasets are less likely to parse the self-training data correctly. For example, the f-score of WSJ data parsed by the base PCFG parser of (Charniak and Johnson, 2005) when trained on the training sections of WSJ is between 89% to 90%, while the f-score of WSJ data parsed with the Collins' model that we use, and a small seed, is between 40% and 80%. As a result, the good results of (McClosky et al, 2006a; 2006b) with large seed sets do not immediately imply success with small seed sets. Demonstration of such success is a contribution of the present paper.

Bacchiani et al (2006) explored the scenario of out-of-domain seed data (the Brown training set containing about 20K sentences) and in-domain self-training data (between 4K to 200K sentences from the WSJ) and showed an improvement over the baseline of training the parser with the seed data only. However, they did not explore the case of small seed datasets (the effort in manually annotating 20K is substantial) and their work addresses only one of our scenarios (OI, see below).

A work closely related to ours is (Steedman et al., 2003a), which applied co-training (Blum and Mitchell, 1998) and self-training to Collins' parsing model using a small seed dataset (500 sentences for both methods and 1,000 sentences for co-training only). The seed, self-training and test datasets they used are similar to those we use in our II experiment (see below), but the self-training protocols are different. They first train the parser with the seed sentences sampled from WSJ sections 2-21. Then, iteratively, 30 sentences are sampled from these sections, parsed by the parser, and the 20 best sentences (in terms of parser confidence defined as probability of top parse) are selected and combined with the previously annotated data to retrain the parser. The co-training protocol is similar except that each parser is trained with the 20 best sentences of the other parser. Self-training did not improve parser performance on the WSJ test section (23). Steedman et al (2003b) followed a similar co-training protocol except that the selection function (three functions were explored) considered the differences between the confidence scores of the two parsers. In this paper we show a self-training protocol that achieves better results than all of these methods (Table 2). The next section discusses possible explanations for the difference in results. Steedman et al (2003b) and Hwa et al, (2003) also used several versions of corrected co-training which are not comparable to ours and other suggested methods because their evaluation requires different measures (e.g. reviewed and corrected constituents are separately counted).

As far as we know, (Becker and Osborne, 2005) is the only additional work that tries to improve a generative PCFG parsers using small seed data. The techniques used are based on active learning (Cohn et al., 1994). The authors test two novel methods, along with the tree entropy (TE) method of (Hwa, 2004). The seed, the unannotated and the test sets, as well as the parser used in that work, are similar to those we use in our II experiment. Our results are superior, as shown in Table 3.

3 Self-Training Protocols

There are many possible ways to do self-training. A main goal of this paper is to identify a self-training protocol most suitable for enhancement and domain adaptation of statistical parsers trained on small datasets. No previous work has succeeded in identifying such a protocol for this task. In this section we try to understand why.

In the protocol we apply, the self-training set contains several thousand sentences A parser trained with a small seed set parses the self-training set, and then the *whole* automatically annotated self-training set is combined with the manually annotated seed set to retrain the parser. This protocol and that of Steedman et al (2003a) were applied to the problem, with the same seed, self-training and test sets. As we show below (see Section 4 and Section 5), while Steedman's protocol does not improve over the baseline of using only the seed data, our protocol does.

There are four differences between the protocols. First, Steedman et al's seed set consists of *consecutive* WSJ sentences, while we select them randomly. In the next section we show that this difference is immaterial. Second, Steedman et al's protocol looks for sentences of high quality parse, while our protocol prefers to use many sentences without checking their parse quality. Third, their protocol is iterative while ours uses a single step. Fourth, our self-training set is orders of magnitude larger than theirs. To examine the parse quality issue, we performed their experiment using their setting but selecting the high quality parse sentences using their f-score relative to the gold standard annotation from secs 2-21 rather than a quality estimate. No improvement over the baseline was achieved even with this oracle. Thus the problem with their protocol does not lie with the parse quality assessment function; no other function would produce results better than the oracle. To examine the iteration issue, we performed their experiment in a single step, selecting at once the oracle-best 2,000 among 3,000 sentences[2], which produced only a mediocre improvement. We thus conclude that the size of the self-training set is a major factor responsible for the difference between the protocols.

4 Experimental Setup

We used a reimplementation of Collins' parsing model 2 (Bikel, 2004). We performed four experiments, II, IO, OI, and OO, two with in-domain seed

[2]Corresponding to a 100 iterations of 30 sentences each.

(II, IO) and two with out-of-domain seed (OI, OO), examining in-domain self-training (II, OI) and out-of-domain self-training (IO, OO). Note that being 'in' or 'out' of domain is determined by the *test* data. Each experiment contained 19 runs. In each run a different seed size was used, from 100 sentences onwards, in steps of 100. For statistical significance, we repeated each experiment five times, in each repetition randomly sampling different manually annotated sentences to form the seed dataset[3].

The seed data were taken from WSJ sections 2-21. For II and IO, the test data is WSJ section 23 (2416 sentences) and the self-training data are either WSJ sections 2-21 (in II, excluding the seed sentences) or the Brown training section (in IO). For OI and OO, the test data is the Brown test section (2424 sentences), and the self-training data is either the Brown training section (in OI) or WSJ sections 2-21 (in OO). We removed the manual annotations from the self-training sections before using them.

For the Brown corpus, we based our division on (Bacchiani et al., 2006; McClosky et al., 2006b). The test and training sections consist of sentences from all of the genres that form the corpus. The training division consists of 90% (9 of each 10 consecutive sentences) of the data, and the test section are the remaining 10% (We did not use any held out data). Parsing performance is measured by f-score, $f = \frac{2 \times P \times R}{P+R}$, where P, R are labeled precision and recall.

To further demonstrate our results for parser adaptation, we also performed the OI experiment where seed data is taken from WSJ sections 2-21 and both self-training and test data are taken from the Switchboard corpus. The distance between the domains of these corpora is much greater than the distance between the domains of WSJ and Brown. The Brown and Switchboard corpora were divided to sections in the same way.

We have also performed all four experiments with the seed data taken from the Brown training section. The results were very similar and will not be detailed here due to space constraints.

[3] (Steedman et al., 2003a) used the *first* 500 sentences of WSJ training section as seed data. For direct comparison, we performed our protocol in the II scenario using the first 500 or 1000 sentences of WSJ training section as seed data and got similar results to those reported below for our protocol with *random* selection. We also applied the protocol of Steedman et al to scenario II with 500 randomly selected sentences, getting no improvement over the random baseline.

5 Results

5.1 In-domain seed data

In these two experiments we show that when the seed and test data are taken from the same domain, a very significant enhancement of parser performance can be achieved, whether the self-training material is in-domain (II) or out-of-domain (IO). Figure 1 shows the improvement in parser f-score when self-training data is used, compared to when it is not used. Table 1 shows the reduction in manually annotated seed data needed to achieve certain f-score levels. The enhancement in performance is very impressive in the in-domain self-training data scenario – a reduction of 50% in the number of manually annotated sentences needed for achieving 75 and 80 f-score values. A significant improvement is achieved in the out-of-domain self-training scenario as well.

Table 2 compares our results with self-training and co-training results reported by (Steedman et al, 20003a; 2003b). As stated earlier, the experimental setup of these works is similar to ours, but the self-training protocols are different. For self-training, our II improves an absolute 3.74% over their 74.3% result, which constitutes a 14.5% reduction in error (from 25.7%).

The table shows that for both seed sizes our self training protocol outperforms both the self-training and co-training protocols of (Steedman et al, 20003a; 2003b). Results are not included in the table only if they are not reported in the relevant paper. The self-training protocol of (Steedman et al., 2003a) does not actually improve over the baseline of using only the seed data. Section 3 discussed a possible explanation to the difference in results.

In Table 3 we compare our results to the results of the methods tested in (Becker and Osborne, 2005) (including TE)[4]. To do that, we compare the reduction in manually annotated data needed to achieve an f-score value of 80 on WSJ section 23 achieved by each method. We chose this measure since it is

[4] The measure is constituents and not sentences because this is how results are reported in (Becker and Osborne, 2005). However, the same reduction is obtained when sentences are counted, because the number of constituents is averaged when taking many sentences.

f-score	75	80
Seed data only	600(0%)	1400(0%)
II	**300**(**50**%)	**700**(**50**%)
IO	500(17%)	1200(14.5%)

Table 1: Number of in-domain seed sentences needed for achieving certain f-scores. Reductions compared to no self-training (line 1) are given in parentheses.

Seed size	our II	our IO	Steedman ST	Steedman CT 2003a	Steedman CT 2003b
500 sent.	**78.04**	75.81	74.3	76.9	—-
1,000 sent.	**81.43**	79.49	—-	79	81.2

Table 2: F-scores of our in-domain-seed self-training vs. self-training (ST) and co-training (CT) of (Steedman et al, 20003a; 2003b).

the only explicitly reported number in that work. As the table shows, our method is superior: our reduction of 50% constitutes an improvement of 66% over their best reduction of 30.6%.

When applying self-training to a parser trained with a small dataset we expect the coverage of the parser to increase, since the combined training set should contain items that the seed dataset does not. On the other hand, since the accuracy of annotation of such a parser is poor (see the no self-training curve in Figure 1) the combined training set surely includes inaccurate labels that might harm parser performance. Figure 2 (left) shows the increase in coverage achieved for in-domain and out-of-domain self-training data. The improvements induced by both methods are similar. This is quite surprising given that the Brown sections we used as self-training data contain science, fiction, humor, romance, mystery and adventure texts while the test section in these experiments, WSJ section 23, contains only news articles.

Figure 2 also compares recall (middle) and precision (right) for the different methods. For II there is a significant improvement in both precision and recall even though many more sentences are parsed. For IO, there is a large gain in recall and a much smaller loss in precision, yielding a substantial improvement in f-score (Figure 1).

F - score	This work - II	Becker unparsed	Becker entropy/unparsed	Hwa TE
80	**50%**	29.4%	30.6%	-5.7%

Table 3: Reduction of the number of manually annotated constituents needed for achieving f score value of 80 on section 23 of the WSJ. In all cases the seed and additional sentences selected to train the parser are taken from sections 02-21 of WSJ.

5.2 Out-of-domain seed data

In these two experiments we show that self-training is valuable for adapting parsers from one domain to another. Figure 3 compares out-of-domain seed data used with in-domain (OI) or out-of-domain (OO) self-training data against the baseline of training only with the out-of-domain seed data.

The left graph shows a significant improvement in f-score. In the middle and right graphs we examine the quality of the parses produced by the model by plotting recall and precision vs. seed size. Regarding precision, the difference between the three conditions is small relative to the f-score difference shown in the left graph. The improvement in the recall measure is much greater than the precision differences, and this is reflected in the f-score result. The gain in coverage achieved by both methods, which is not shown in the figure, is similar to that reported for the in-domain seed experiments. The left graph along with the increase in coverage show the power of self-training in parser adaptation when small seed datasets are used: not only do OO and OI parse many more sentences than the baseline, but their f-score values are consistently better.

To see how much manually annotated data can be saved by using out-of-domain seed, we train the parsing model with manually annotated data from the Brown training section, as described in Section 4. We assume that given a fixed number of training sentences the best performance of the parser without self-training will occur when these sentences are selected from the domain of the test section, the Brown corpus. We compare the amounts of manually annotated data needed to achieve certain f-score levels in this condition with the corresponding amounts of data needed by OI and OO. The results are summarized in Table 4. We compare to two baselines using in- and out-of-domain seed data without

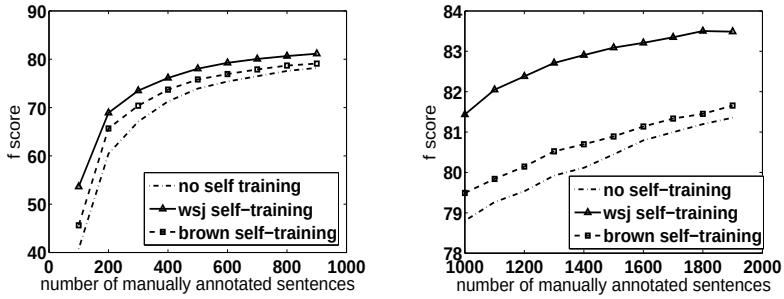


Figure 1: Number of seed sentences vs. f-score, for the two in-domain seed experiments: II (triangles) and IO (squares), and for the no self-training baseline. Self-training provides a substantial improvement.

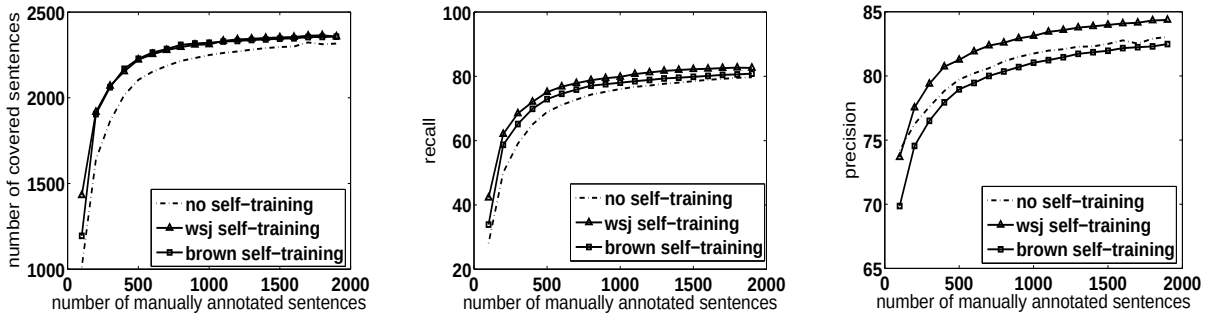


Figure 2: Number of seed sentences vs. coverage (left), recall (middle) and precision (right) for the two in-domain seed experiments: II (triangles) and IO (squares), and for the no self-training baseline.

any self-training. The second line (ID) serves as a reference to compute how much manual annotation of the test domain was saved, and the first line (OD) serves as a reference to show by how much self-training improves the out-of-domain baseline. The table stops at an f-score of 74 because that is the best that the baselines can do.

A significant reduction in annotation cost over the ID baseline is achieved where the seed size is between 100 and 1200. Improvement over the OD baseline is for the whole range of seed sizes. Both OO and OI achieve 20-33% reduction in manual annotation compared to the ID baseline and enhance the performance of the parser by as much as 42.9%.

The only previous work that adapts a parser trained on a small dataset between domains is that of (Steedman et al., 2003a), which used co-training (no self-training results were reported there or elsewhere). In order to compare with that work, we performed OI with seed taken from the Brown corpus and self-training and test taken from WSJ, which is the setup they use, obtaining a similar improvement to that reported there. However, co-training is a more complex method that requires an additional parser (LTAG in their case).

To further substantiate our results for the parser adaptation scenario, we used an additional corpus, Switchboard. Figure 4 shows the results of an OI experiment with WSJ seed and Switchboard self-training and test data. Although the domains of these two corpora are very different (more so than WSJ and Brown), self-training provides a substantial improvement.

We have also performed all four experiments with Brown and WSJ trading places. The results obtained were very similar to those reported here, and will not be detailed due to lack of space.

6 Analysis

In this section we try to better understand the benefit in using self-training with small seed datasets. We formulate the following criterion: the number of words in a test sentence that do not appear in the seed data ('unknown words') is a strong indicator

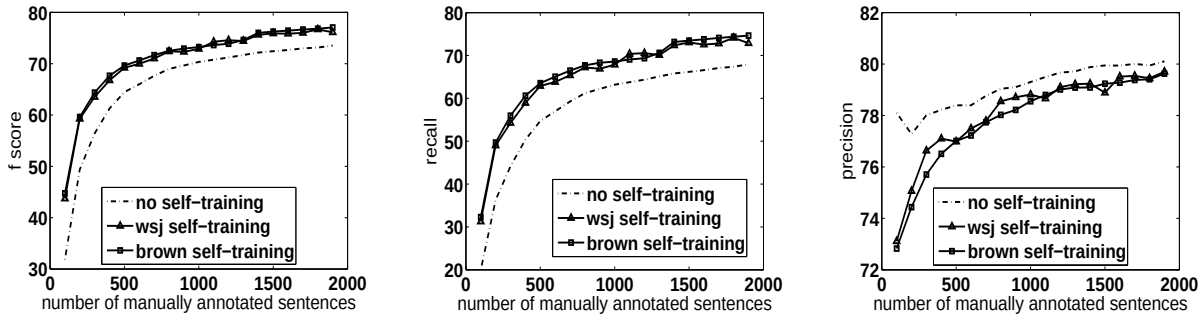


Figure 3: Number of seed sentences vs. f-score (left), recall (middle) and precision (right), for the two out-of-domain seed data experiments: OO (triangles) and OI (squares), and for the no self-training baseline.

f-sc.	66	68	70	72	74
OD	600	800	1,000	1,400	–
ID	600	700	800	1,000	1,200
OO	**400**	**500**	**600**	**800**	**1100**
	33, 33	28.6, 37.5	33, 40	20, 42.9	8, –
OI	**400**	**500**	**600**	**800**	1,300
	33, 33	28.6, 37.5	33, 40	20, 42.9	−8, –

Table 4: Number of manually annotated seed sentences needed for achieving certain f-score values. The first two lines show the out-of-domain and in-domain seed baselines. The reductions compared to the baselines is given as ID, OD.

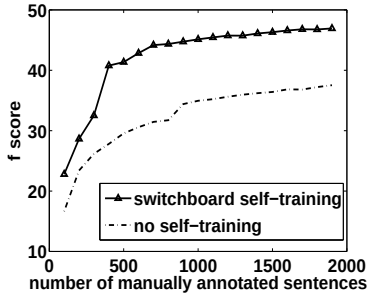


Figure 4: Number of seed sentences vs. f-score, for the OI experiment using WSJ seed data and SwitchBoard self-training and test data. In spite of the strong dissimilarity between the domains, self-training provides a substantial improvement.

to whether it is worthwhile to use small seed self-training. Figure 5 shows the number of unknown words in a sentence vs. the probability that the self-training model will parse a sentence no worse (upper curve) or better (lower curve) than the baseline model.

The upper curve shows that regardless of the

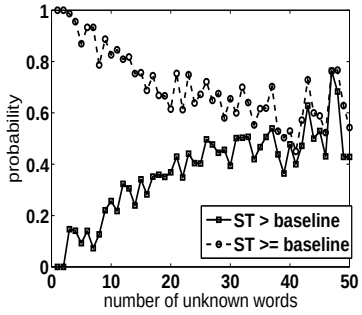


Figure 5: For sentences having the same number of unknown words, we show the probability that the self-training model parses a sentence from the set no worse (upper curve) or better (lower curve) than the baseline model.

number of unknown words in the sentence, there is more than 50% chance that the self-training model will not harm the result. This probability decreases from almost 1 for a very small number of unknown words to about 0.55 for 50 unknown words. The lower curve shows that when the number of unknown words increases, the probability that the self-training model will do better than the baseline model increases from almost 0 (for a very small number of unknown words) to about 0.55. Hence, the number of unknown words is an indication for the potential benefit (value on the lower curve) and risk (1 minus the value on the upper curve) in using the self-training model compared to using the baseline model. Unknown words were not identified in (McClosky et al., 2006a) as a useful predictor for the benefit of self-training.

We also identified a length effect similar to that studied by (McClosky et al., 2006a) for self-training (using a reranker and large seed, as detailed in Section 2). Due to space limitations we do not discuss it here.

7 Discussion

Self-training is usually not considered to be a valuable technique in improving the performance of generative statistical parsers, especially when the manually annotated seed sentence dataset is small. Indeed, in the II scenario, (Steedman et al., 2003a; McClosky et al., 2006a; Charniak, 1997) reported no improvement of the base parser for small (500 sentences, in the first paper) and large (40K sentences, in the last two papers) seed datasets respectively. In the II, OO, and OI scenarios, (McClosky et al, 2006a; 2006b) succeeded in improving the parser performance only when a reranker was used to reorder the 50-best list of the generative parser, with a seed size of 40K sentences. Bacchiani et al (2006) improved the parser performance in the OI scenario but their seed size was large (about 20K sentences).

In this paper we have shown that self-training can enhance the performance of generative parsers, without a reranker, in four in- and out-of-domain scenarios using a small seed dataset. For the II, IO and OO scenarios, we are the first to show improvement by self-training for generative parsers. We achieved a 50% (20-33%) reduction in annotation cost for the in-domain (out-of-domain) seed data scenarios. Previous work with small seed datasets considered only the II and OI scenarios. Our results for the former are better than any previous method, and our results for the latter (which are the first reported self-training results) are similar to previous results for co-training, a more complex method. We demonstrated our results using three corpora of varying degrees of domain difference.

A direction for future research is combining self-training data from various domains to enhance parser adaptation.

Acknowledgement. We would like to thank Dan Roth for his constructive comments on this paper.

References

Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat, 2006. MAP adaptation of stochastic grammars. *Computer Speech and Language*, 20(1):41–68.

Markus Becker and Miles Osborne, 2005. A two-stage method for active learning of statistical grammars. *IJCAI '05*.

Daniel Bikel, 2004. *Code developed at University of Pennsylvania*. http://www.cis.upenn.edu.bikel.

Avrim Blum and Tom M. Mitchell, 1998. Combining labeled and unlabeled data with co-training. *COLT '98*.

Eugene Charniak, 1997. Statistical parsing with a context-free grammar and word statistics. *AAAI '97*.

Eugene Charniak, 2000. A maximum-entropy-inspired parser. *ANLP '00*.

Eugene Charniak and Mark Johnson, 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. *ACL '05*.

Stephen Clark, James Curran, and Miles Osborne, 2003. Bootstrapping pos taggers using unlabelled data. *CoNLL '03*.

David A. Cohn, Les Atlas, and Richard E. Ladner, 1994. Improving generalization with active learning. *Machine Learning*, 15(2):201–221.

Michael Collins, 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania.

Rebecca Hwa, Miles Osborne, Anoop Sarkar and Mark Steedman, 2003. Corrected co-training for statistical parsers. In *ICML '03, Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining.*

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

Terry Koo and Michael Collins, 2005. Hidden-variable models for discriminative reranking. *EMNLP '05*.

David McClosky, Eugene Charniak, and Mark Johnson, 2006a. Effective self-training for parsing. *HLT-NAACL '06*.

David McClosky, Eugene Charniak, and Mark Johnson, 2006b. Reranking and self-training for parser adaptation. *ACL-COLING '06*.

Mark Steedman, Anoop Sarkar, Miles Osborne, Rebecca Hwa, Stephen Clark, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim, 2003a. Bootstrapping statistical parsers from small datasets. *EACL '03*.

Mark Steedman, Rebecca Hwa, Stephen Clark, Miles Osborne, Anoop Sarkar, Julia Hockenmaier, Paul Ruhlen,Steven Baker, and Jeremiah Crim, 2003b. Example selection for bootstrapping statistical parsers. *NAACL '03*.

HPSG Parsing with Shallow Dependency Constraints

Kenji Sagae[1] and **Yusuke Miyao**[1] and **Jun'ichi Tsujii**[1,2,3]
[1]Department of Computer Science
University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo, Japan
[2]School of Computer Science, University of Manchester
[3]National Center for Text Mining
{sagae,yusuke,tsujii}@is.s.u-tokyo.ac.jp

Abstract

We present a novel framework that combines strengths from surface syntactic parsing and deep syntactic parsing to increase deep parsing accuracy, specifically by combining dependency and HPSG parsing. We show that by using surface dependencies to constrain the application of wide-coverage HPSG rules, we can benefit from a number of parsing techniques designed for high-accuracy dependency parsing, while actually performing deep syntactic analysis. Our framework results in a 1.4% absolute improvement over a state-of-the-art approach for wide coverage HPSG parsing.

1 Introduction

Several efficient, accurate and robust approaches to data-driven dependency parsing have been proposed recently (Nivre and Scholz, 2004; McDonald et al., 2005; Buchholz and Marsi, 2006) for syntactic analysis of natural language using bilexical dependency relations (Eisner, 1996). Much of the appeal of these approaches is tied to the use of a simple formalism, which allows for the use of efficient parsing algorithms, as well as straightforward ways to train discriminative models to perform disambiguation. At the same time, there is growing interest in parsing with more sophisticated lexicalized grammar formalisms, such as Lexical Functional Grammar (LFG) (Bresnan, 1982), Lexicalized Tree Adjoining Grammar (LTAG) (Schabes et al., 1988), Head-driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994) and Combinatory Categorial Grammar (CCG) (Steedman, 2000), which represent deep syntactic structures that cannot be expressed in a shallower formalism designed to represent only aspects of surface syntax, such as the dependency formalism used in current mainstream dependency parsing.

We present a novel framework that combines strengths from surface syntactic parsing and deep syntactic parsing, specifically by combining dependency and HPSG parsing. We show that, by using surface dependencies to constrain the application of wide-coverage HPSG rules, we can benefit from a number of parsing techniques designed for high-accuracy dependency parsing, while actually performing deep syntactic analysis. From the point of view of HPSG parsing, accuracy can be improved significantly through the use of highly accurate discriminative dependency models, without the difficulties involved in adapting these models to a more complex and linguistically sophisticated formalism. In addition, improvements in dependency parsing accuracy are converted directly into improvements in HPSG parsing accuracy. From the point of view of dependency parsing, the application of HPSG rules to structures generated by a surface dependency model provides a principled and linguistically motivated way to identify deep syntactic phenomena, such as long-distance dependencies, raising and control.

We begin by describing our dependency and HPSG parsing approaches in section 2. In section 3, we present our framework for HPSG parsing with shallow dependency constraints, and in section 4 we

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 624–631,
Prague, Czech Republic, June 2007.

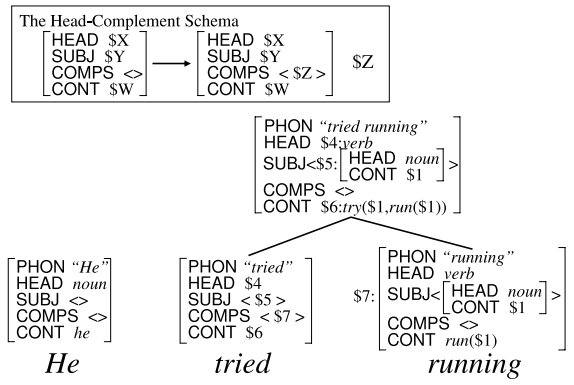


Figure 1: HPSG parsing

evaluate this framework empirically. Sections 5 and 6 discuss related work and conclusions.

2 Fast dependency parsing and wide-coverage HPSG parsing

2.1 Data-driven dependency parsing

Because we use dependency parsing as a step in deep parsing, it is important that we choose a parsing approach that is not only accurate, but also efficient. The deterministic shift/reduce classifier-based dependency parsing approach (Nivre and Scholz, 2004) has been shown to offer state-of-the-art accuracy (Nivre et al., 2006) with high efficiency due to a greedy search strategy. Our approach is based on Nivre and Scholz's approach, using support vector machines for classification of shift/reduce actions.

2.2 Wide-coverage HPSG parsing

HPSG (Pollard and Sag, 1994) is a syntactic theory based on lexicalized grammar formalism. In HPSG, a small number of *schemas* explain general construction rules, and a large number of *lexical entries* express word-specific syntactic/semantic constraints. Figure 1 shows an example of the process of HPSG parsing. First, lexical entries are assigned to each word in a sentence. In Figure 1, lexical entries express subcategorization frames and predicate argument structures. Parsing proceeds by applying schemas to lexical entries. In this example, the Head-Complement Schema is applied to the lexical entries of "*tried*" and "*running*". We then obtain a phrasal structure for "*tried running*". By repeatedly applying schemas to lexical/phrasal structures,

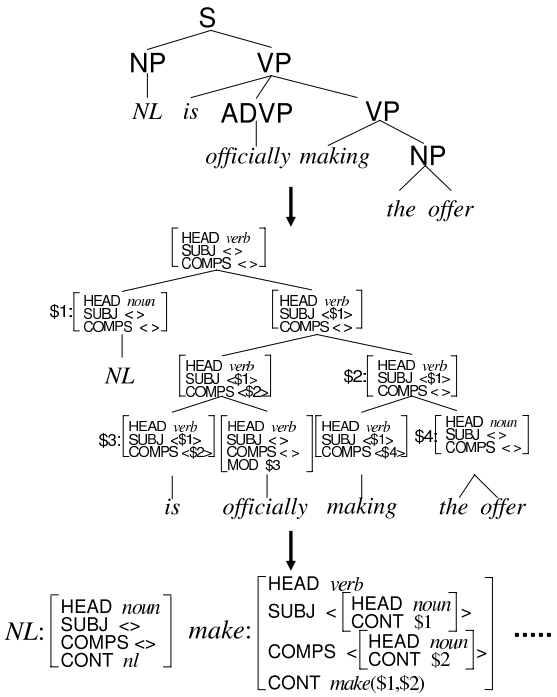


Figure 2: Extracting HPSG lexical entries from the Penn Treebank

we finally obtain an HPSG parse tree that covers the entire sentence.

In this paper, we use an HPSG parser developed by Miyao and Tsujii (2005). This parser has a wide-coverage HPSG lexicon which is extracted from the Penn Treebank. Figure 2 illustrates their method for extraction of HPSG lexical entries. First, given a parse tree from the Penn Treebank (top), HPSG-style constraints are added and an HPSG-style parse tree is obtained (middle). Lexical entries are then extracted from the terminal nodes of the HPSG parse tree (bottom). This way, in addition to a wide-coverage lexicon, we also obtain an HPSG treebank, which can be used as training data for disambiguation models.

The disambiguation model of this parser is based on a maximum entropy model (Berger et al., 1996). The probability $p(T|W)$ of an HPSG parse tree T for the sentence $W = \langle w_1, \ldots, w_n \rangle$ is given as:

$$\begin{aligned} p(T|W) &= p(T|L,W)p(L|W) \\ &= \frac{1}{Z}\exp\left(\sum_i \lambda_i f_i(T)\right)\prod_j p(l_j|W), \end{aligned}$$

where $L = \langle l_1, \ldots, l_n \rangle$ are lexical entries and

$p(l_i|W)$ is the *supertagging probability*, i.e., the probability of assignining the lexical entry l_i to w_i (Ninomiya et al., 2006). The probability $p(T|L, W)$ is a maximum entropy model on HPSG parse trees, where Z is a normalization factor, and feature functions $f_i(T)$ represent syntactic characteristics, such as head words, lengths of phrases, and applied schemas. Given the HPSG treebank as training data, the model parameters λ_i are estimated so as to maximize the log-likelihood of the training data (Malouf, 2002).

3 HPSG parsing with dependency constraints

While a number of fairly straightforward models can be applied successfully to dependency parsing, designing and training HPSG parsing models has been regarded as a significantly more complex task. Although it seems intuitive that a more sophisticated linguistic formalism should be more difficult to parameterize properly, we argue that the difference in complexity between HPSG and dependency structures can be seen as incremental, and that the use of accurate and efficient techniques to determine the surface dependency structure of a sentence provides valuable information that aids HPSG disambiguation. This is largely because HPSG is based on a lexicalized grammar formalism, and as such its syntactic structures have an underlying dependency backbone. However, HPSG syntactic structures includes long-distance dependencies, and the underlying dependency structure described by and HPSG structure is a directed acyclic graph, not a dependency tree (as used by mainstream approaches to data-driven dependency parsing). This difference manifests itself in words that have multiple heads. For example, in the sentence *I tried to run*, the pronoun *I* is a dependent of *tried* and of *run*. This makes it possible to represent that *I* is the subject of both verbs, precisely the kind of information that cannot be represented in dependency parsing. If we ignore long-distance dependencies, however, HPSG structures can be seen as lexicalized trees that can be easily converted into dependency trees.

Given that for an HPSG representation of the syntactic structure of a sentence we can determine a dependency tree by removing long-distance dependencies, we can use dependency parsing techniques (such as the deterministic dependency parsing approach mentioned in section 2.1) to determine the underlying dependency trees in HPSG structures. This is the basis for the parsing framework presented here. In this approach, deep dependency analysis is done in two stages. First, a dependency parser determines the shallow dependency tree for the input sentence. This shallow dependency tree corresponds to the underlying dependency graph of the HPSG structure for the input sentence, without dependencies that roughly correspond to deep syntax. The second step is to perform HPSG parsing, as described in section 2.2, but using the shallow dependency tree to constrain the application of HPSG rules. We now discuss these two steps in more detail.

3.1 Determining shallow dependencies in HPSG structures using dependency parsing

In order to apply a data-driven dependency approach to the task of identifying the shallow dependency tree in HPSG structures, we first need a corpus of such dependency trees to serve as training data. We created a dependency training corpus based on the Penn Treebank (Marcus et al., 1993), or more specifically on the HPSG Treebank generated from the Penn Treebank (see section 2.2). For each HPSG structure in the HPSG Treebank, a dependency tree is extracted in two steps. First, the HPSG tree is converted into a CFG-style tree, simply by removing long-distance dependency links between nodes. A dependency tree is then extracted from the resulting lexicalized CFG-style tree, as is commonly done for converting constituent trees into dependency trees after the application of a head-percolation table (Collins, 1999).

Once a dependency training corpus is available, it is used to train a dependency parser as described in section 2.1. This is done by training a classifier to determine parser actions based on local features that represent the current state of the parser (Nivre and Scholz, 2004; Sagae and Lavie, 2005). Training data for the classifier is obtained by applying the parsing algorithm over the training sentences (for which the correct dependency structures are known) and recording the appropriate parser actions that result in the formation of the correct dependency trees, coupled with the features that represent the state of

the parser mentioned in section 2.1. An evaluation of the resulting dependency parser and its efficacy in aiding HPSG parsing is presented in section 4.

3.2 Parsing with dependency constraints

Given a set of dependencies, the bottom-up process of HPSG parsing can be constrained so that it does not violate the given dependencies. This can be achieved by a simple extension of the parsing algorithm, as follows. During parsing, we store the lexical head of each partial parse tree. In each schema application, we can determine which child is the head; for example, the left child is the head when we apply the Head-Complement Schema. Given this information and lexical heads, the parser can identify the dependency produced by this schema application, and can therefore judge whether the schema application violates the dependency constraints.

This method forces the HPSG parser to produce parse trees that strictly conform to the output of the dependency parser. However, this means that the HPSG parser outputs no successful parse results when it cannot find the parse tree that is completely consistent with the given dependencies. This situation may occur when the dependency parser produces structures that are not covered in the HPSG grammar. This is especially likely with a fully data-driven dependency parser that uses local classification, since its output may not be globally consistent grammatically. In addition, the HPSG grammar is extracted from the HPSG Treebank using a corpus-based procedure, and it does not necessarily cover all possible grammatical phenomena in unseen text (Miyao and Tsujii, 2005).

We therefore propose an extension of this approach that uses predetermined dependencies as *soft* constraints. Violations of schema applications are detected in the same way as before, but instead of strictly prohibiting schema applications, we penalize the log-likelihood of partial parse trees created by schema applications that violate the dependencies constraints. Given a negative value α, we add α to the log-probability of a partial parse tree when the schema application violates the dependency constraints. That is, when a parse tree violates n dependencies, the log-probability of the parse tree is lowered by $n\alpha$. The meta parameter α is determined so as to maximize the accuracy on the development set.

Soft dependency constraints can be implemented as explained above as a straightforward extension of the parsing algorithm. In addition, it is easily integrated with beam thresholding methods of parsing. Because beam thresholding discards partial parse trees that have low log-probabilities, we can expect that the parser would discard partial parse trees based on violation of the dependency constraints.

4 Experiments

We evaluate the accuracy of HPSG parsing with dependency constraints on the HPSG Treebank (Miyao et al., 2003), which is extracted from the Wall Street Journal portion of the Penn Treebank (Marcus et al., 1993)[1]. Sections 02-21 were used for training (for HPSG and dependency parsers), section 22 was used as development data, and final testing was performed on section 23. Following previous work on wide-coverage parsing with lexicalized grammars using the Penn Treebank, we evaluate the parser by measuring the accuracy of predicate-argument relations in the parser's output. A predicate-argument relation is defined as a tuple $\langle \sigma, w_h, a, w_a \rangle$, where σ is the predicate type (e.g. adjective, intransitive verb), w_h is the head word of the predicate, a is the argument label (MODARG, ARG1, ... , ARG4), and w_a is the head word of the argument. Labeled precision (LP)/labeled recall (LR) is the ratio of tuples correctly identified by the parser. These predicate-argument relations cover the full range of syntactic dependencies produced by the HPSG parser (including, long-distance dependencies, raising and control, in addition to surface dependencies).

In the experiments presented in this section, input sentences were automatically tagged with parts-of-speech with about 97% accuracy, using a maximum entropy POS tagger. We also report results on parsing text with gold standard POS tags, where explicitly noted. This provides an upper-bound on what can be expected if a more sophisticated multi-tagging scheme (James R. Curran and Vadas, 2006) is used, instead of hard assignment of single tags in a preprocessing step as done here.

[1] The extraction software can be obtained from http://www-tsujii.is.s.u-tokyo.ac.jp/enju.

4.1 Baseline

HPSG parsing results using the same HPSG grammar and treebank have recently been reported by Miyao and Tsujii (2005) and Ninomia et al. (2006). By running the HPSG parser described in section 2.2 on the development data *without* dependency constraints, we obtain similar values of LP (86.8%) and LR (85.6%) as those reported by Miyao and Tsujii (Miyao and Tsujii, 2005). Using the extremely lexicalized framework of (Ninomiya et al., 2006) by performing supertagging before parsing, we obtain similar accuracy as Ninomiya et al. (87.1% LP and 85.9% LR).

4.2 Dependency constraints and the penalty parameter

Parsing the development data with *hard dependency constraints* confirmed the intuition that these constraints often describe dependency structures that do not conform to HPSG schema used in parsing, resulting in parse failures. To determine the upper-bound on HPSG parsing with hard dependency constraints, we set the HPSG parser to disallow the application of any rules that result in the creation of dependencies that violate *gold standard dependencies*. This results in high precision (96.7%), but recall is low (82.3%) due to parse failures caused by lack of grammatical coverage [2]. Using dependencies produced by the shift-reduce SVM parser, we obtain 91.5% LP and 65.7% LR. This represents a large gain in precision over the baseline, but an even greater loss in recall, which limits the usefulness of the parser, and severely hurts the appeal of hard constraints.

We focus the rest of our experiments on parsing with *soft dependency constraints*. As explained in section 3, this involves setting the penalty parameter α. During parsing, we subtract α from the log-probability of applying any schema that violates the dependency constraints given to the HPSG parser. Figure 3 illustrates the effect of α when gold standard dependencies (and gold standard POS tags) are used. We note that setting $\alpha = 0$ causes the parser to ignore dependency constraints, providing baseline performance. Conversely, setting a high enough value ($\alpha = 30$ is sufficient, in practice) causes any substructures that violate the dependency constraints to be used only when they are absolutely necessary to produce a valid parse for the input sentence. In figure 3, this corresponds to an upper-bound on the accuracy of parsing with soft dependency constraints (94.7% f-score), since gold standard dependencies are used.

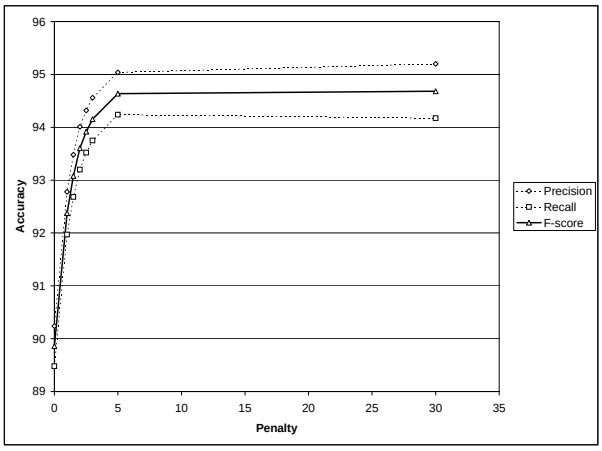


Figure 3: The effect of α on HPSG parsing constrained by gold standard dependencies.

We set α empirically with simple hill climbing on the development set. Because it is expected that the optimal value of α depends on the accuracy of the surface dependency parser, we set separate values for parsing with a POS tagger or with gold standard POS tags. Figure 4 shows the accuracy of HPSG predicate-argument relations obtained with dependency constraints determined by dependency parsing with gold standard POS tags. With both automatically assigned and gold standard POS tags, we observe an improvement of about 0.6% in precision, recall and f-score, when the optimal α value is used in each case. While this corresponds to a relative error reduction of over 6% (or 12%, if we consider the upper-bound dictated by imperfect grammatical coverage), a more interesting aspect of this framework is that it allows techniques designed for improving dependency accuracy to improve HPSG parsing accuracy directly, as we illustrate next.

[2] Although the HPSG grammar does not have perfect coverage of unseen text, it supports complete and *mostly* correct analyses for all sentences in the development set. However, when we require *completely* correct analyses by using hard constraints, lack of coverage may cause parse failures.

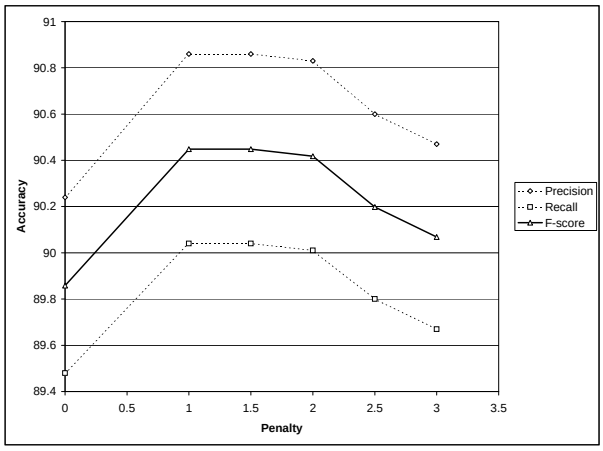


Figure 4: The effect of α on HPSG parsing constrained by the output of a dependency parser using gold standard POS tags.

4.3 Determining constraints with dependency parser combination

Parser combination has been shown to be a powerful way to obtain very high accuracy in dependency parsing (Sagae and Lavie, 2006). Using dependency constraints allows us to improve HPSG parsing accuracy simply by using an existing parser combination approach. As a first step, we train two additional parsers with the dependencies extracted from the HPSG Treebank. The first uses the same shift-reduce framework described in section 2.1, but it process the input from right to left (RL). This has been found to work well in previous work on dependency parser combination (Zeman and Žabokrtský, 2005; Sagae and Lavie, 2006). The second parser is MSTParser, the large-margin maximum spanning tree parser described in (McDonald et al., 2005)[3].

We examine the use of two combination schemes: one using two parsers, and one using three parsers. The first combination approach is to keep only dependencies for which there is agreement between the two parsers. In other words, dependencies that are proposed by one parser but not the other are simply discarded. Using the left-to-right shift-reduce parser and MSTParser, we find that this results in very high precision of surface dependencies on the development data. In the second approach, combination of the three dependency parsers is done according to the maximum spanning tree combination scheme of Sagae and Lavie (2006), which results in high accuracy of surface dependencies. For each of the combination approaches, we use the resulting dependencies as constraints for HPSG parsing, determining the optimal value of α on the development set in the same way as done for a single parser. Table 1 summarizes our experiments on development data using parser combinations to produce dependency constraints [4]. The two combination approaches are denoted as C1 and C2.

Parser	Dep	α	HPSG	Diff
none (baseline)	–	–	86.5	–
LR shift-reduce	91.2	1.5	87.1	0.6
RL shift-reduce	90.1	–	–	
MSTParser	91.0	–	–	
C1 (agreement)	96.8*	2.5	87.4	0.9
C2 (MST)	92.4	2.5	87.4	0.9

Table 1: Summary of results on development data. * The shallow accuracy of combination C1 corresponds to the dependency precision (no dependencies were reported for 8% of all words in the development set).

4.4 Results

Having determined α values on development data for the shift-reduce dependency parser, the two-parser agreement combination, and the three-parser maximum spanning tree combination, we parse the test data (section 23) using these three different sources of dependency constraints for HPSG parsing. Our final results are shown in table 2, where we also include the results published in (Ninomiya et al., 2006) for comparison purposes, and the result of using dependency constraints obtained with gold standard POS tags.

By using two unlabeled dependency parsers to provide soft dependency constraints, we obtain a 1% absolute improvement in precision and recall of predicate-argument identification in HPSG parsing over a strong baseline. Our baseline approach outperformed previously published results on this test

[3]Downloaded from http://sourceforge.net/projects/mstparser

[4]The accuracy figures for the dependency parsers is expressed as unlabeled accuracy of the surface dependencies only, and are not comparable to the HPSG parsing accuracy figures

Parser	LP	LR	F-score
HPSG Baseline	87.4	87.0	87.2
Shift-Reduce + HPSG	88.2	87.7	87.9
C1 + HPSG	**88.5**	**88.0**	**88.2**
C2 + HPSG	88.4	87.9	88.1
Baseline(gold)	89.8	89.4	89.6
Shift-Reduce(gold)	90.62	90.23	90.42
C1+HPSG(gold)	**90.9**	**90.4**	**90.6**
C2+HPSG(gold)	90.8	90.4	90.6
Miyao and Tsujii, 2005	85.0	84.3	84.6
Ninomiya et al., 2006	87.4	86.3	86.8

Table 2: Final results on test set. The first set of results show our HPSG baseline and HPSG with soft dependency constraints using three different sources of dependency constraints. The second set of results show the accuracy of the same parsers when gold part-of-speech tags are used. The third set of results is from existing published models on the same data.

set, and our best performing combination scheme obtains an absolute improvement of 1.4% over the best previously published results using the HPSG Treebank. It is interesting to note that the results obtained with dependency parser combinations C1 and C2 were very similar, even though in C1 only two parsers were used, and constraints were provided for about 92% of shallow dependencies (with accuracy higher than 96%). Clearly, precision is crucial in dependency constraints.

Finally, although it is necessary to perform dependency parsing to pre-compute dependency constraints, the total time required to perform the entire process of HPSG parsing with dependency constraints is close to that of the baseline HPSG approach. This is due to two reasons: (1) the dependency parsing approaches used to pre-compute constraints are several times faster than the baseline HPSG approach, and (2) the HPSG portion of the process is significantly faster when dependency constraints are used, since the constraints help sharpen the search space, making search more efficient. Using the baseline HPSG approach, it takes approximately 25 minutes to parse the test set. The total time required to parse the test set using HPSG with dependency constraints generated by the shift-reduce parser is 27 minutes. With combination C1, parsing time increases to 30 minutes, since two dependency parsers are used sequentially.

5 Related work

There are other approaches that combine shallow processing with deep parsing (Crysmann et al., 2002; Frank et al., 2003; Daum et al., 2003) to improve parsing *efficiency*. Typically, shallow parsing is used to create robust minimal recursion semantics, which are used as constraints to limit ambiguity during parsing. Our approach, in contrast, uses syntactic dependencies to achieve a significant improvement in the *accuracy* of wide-coverage HPSG parsing. Additionally, our approach is in many ways similar to supertagging (Bangalore and Joshi, 1999), which uses sequence labeling techniques as an efficient way to pre-compute parsing constraints (specifically, the assignment of lexical entries to input words).

6 Conclusion

We have presented a novel framework for taking advantage of the strengths of a shallow parsing approach and a deep parsing approach. We have shown that by constraining the application of rules in HPSG parsing according to results from a dependency parser, we can significantly improve the accuracy of deep parsing by using shallow syntactic analyses.

To illustrate how this framework allows for improvements in the accuracy of dependency parsing to be used directly to improve the accuracy of HPSG parsing, we showed that by combining the results of different dependency parsers using the search-based parsing ensemble approach of (Sagae and Lavie, 2006), we obtain improved HPSG parsing accuracy as a result of the improved dependency accuracy.

Although we have focused on the use of HPSG and dependency parsing, the general framework presented here can be applied to other lexicalized grammar formalisms, such as LTAG, CCG and LFG.

Acknowledgements

This research was partially supported by Grant-in-Aid for Specially Promoted Research 18002007.

References

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

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

Joan Bresnan. 1982. *The mental representation of grammatical relations*. MIT Press.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Natural Language Learning*. New York, NY.

M. Collins. 1999. *Head-Driven Models for Natural Language Parsing*. Phd thesis, University of Pennsylvania.

Berthold Crysmann, Anette Frank, Bernd Kiefer, Stefan Mueller, Guenter Neumann, Jakub Piskorski, Ulrich Schaefer, Melanie Siegel, Hans Uszkoreit, Feiyu Xu, Markus Becker, and Hans-Ulrich Krieger. 2002. An integrated architecture for shallow and deep processing. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*.

Michael Daum, Kilian A. Foth, and Wolfgang Menzel. 2003. Constraint-based integration of deep and shallow parsing techniques. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2003)*.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the International Conference on Computational Linguistics (COLING'96)*. Copenhagen, Denmark.

Anette Frank, Markus Becker, Berthold Crysmann, Bernd Kiefer, and Ulrich Schaefer. 2003. Integrated shallow and deep parsing: TopP meets HPSG. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, pages 104–111.

Stephen Clark James R. Curran and David Vadas. 2006. Multi-tagging for lexicalized-grammar parsing. In *Proceedings of COLING/ACL 2006*. Sydney, Australia.

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 2002 Conference on Natural Language Learning*.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewics. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19.

Ryan McDonald, Fernando Pereira, K. Ribarov, and J. Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technologies/Empirical Methods in Natural Language Processing (HLT-EMNLP)*. Vancouver, Canada.

Yusuke Miyao and Jun'ichi Tsujii. 2005. Probabilistic disambiguation models for wide-coverage hpsg parsing. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*. Ann Arbor, MI.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2003. Corpus oriented grammar development for aquiring a head-driven phrase structure grammar from the penn treebank. In *Proceedings of the Tenth Conference on Natural Language Learning*.

T. Ninomiya, T. Matsuzaki, Y. Tsuruoka, Y. Miyao, and J. Tsujii. 2006. Extremely lexicalized models for accurate and fast hpsg parsing. In *Proceedings of the 2006 Conference on Empirical Methods for Natural Language Processing (EMNLP 2006)*.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of english text. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 64–70. Geneva, Switzerland.

J. Nivre, J. Hall, J. Nilsson, G. Eryigit, and S. Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Natural Language Learning*. New York, NY.

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

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the Ninth International Workshop on Parsing Technologies.* Vancouver, BC.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the 2006 Meeting of the North American ACL.* New York, NY.

Yves Schabes, Anne Abeille, and Aravind Joshi. 1988. Parsing strategies with lexicalized grammars: Application to tree adjoining grammars. In *Proceedings of 12th COLING.*

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

Daniel Zeman and Zdenek Žabokrtský. 2005. Improving parsing accuracy by combining diverse dependency parsers. In *Proceedings of the International Workshop on Parsing Technologies*. Vancouver, Canada.

Constituent Parsing with Incremental Sigmoid Belief Networks

Ivan Titov
Department of Computer Science
University of Geneva
24, rue Général Dufour
CH-1211 Genève 4, Switzerland
ivan.titov@cui.unige.ch

James Henderson
School of Informatics
University of Edinburgh
2 Buccleuch Place
Edinburgh EH8 9LW, United Kingdom
james.henderson@ed.ac.uk

Abstract

We introduce a framework for syntactic parsing with latent variables based on a form of dynamic Sigmoid Belief Networks called Incremental Sigmoid Belief Networks. We demonstrate that a previous feed-forward neural network parsing model can be viewed as a coarse approximation to inference with this class of graphical model. By constructing a more accurate but still tractable approximation, we significantly improve parsing accuracy, suggesting that ISBNs provide a good idealization for parsing. This generative model of parsing achieves state-of-the-art results on WSJ text and 8% error reduction over the baseline neural network parser.

1 Introduction

Latent variable models have recently been of increasing interest in Natural Language Processing, and in parsing in particular (e.g. (Koo and Collins, 2005; Matsuzaki et al., 2005; Riezler et al., 2002)). Latent variables provide a principled way to include features in a probability model without needing to have data labeled with those features in advance. Instead, a labeling with these features can be induced as part of the training process. The difficulty with latent variable models is that even small numbers of latent variables can lead to computationally intractable inference (a.k.a. decoding, parsing). In this paper we propose a solution to this problem based on dynamic Sigmoid Belief Networks (SBNs) (Neal, 1992). The dynamic SBNs which we peopose, called Incremental Sigmoid Belief Networks (ISBNs) have large numbers of latent variables, which makes exact inference intractable. However, they can be approximated sufficiently well to build fast and accurate statistical parsers which induce features during training.

We use SBNs in a generative history-based model of constituent structure parsing. The probability of an unbounded structure is decomposed into a sequence of probabilities for individual derivation decisions, each decision conditioned on the unbounded history of previous decisions. The most common approach to handling the unbounded nature of the histories is to choose a pre-defined set of features which can be unambiguously derived from the history (e.g. (Charniak, 2000; Collins, 1999)). Decision probabilities are then assumed to be independent of all information not represented by this finite set of features. Another previous approach is to use neural networks to compute a compressed representation of the history and condition decisions on this representation (Henderson, 2003; Henderson, 2004). It is possible that an unbounded amount of information is encoded in the compressed representation via its continuous values, but it is not clear whether this is actually happening due to the lack of any principled interpretation for these continuous values.

Like the former approach, we assume that there are a finite set of features which encode the relevant information about the parse history. But unlike that approach, we allow feature values to be ambiguous, and represent each feature as a distribution over (binary) values. In other words, these history features are treated as latent variables. Unfortunately, inter-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 632–639,
Prague, Czech Republic, June 2007.

preting the history representations as distributions over discrete values of latent variables makes the exact computation of decision probabilities intractable. Exact computation requires marginalizing out the latent variables, which involves summing over all possible vectors of discrete values, which is exponential in the length of the vector.

We propose two forms of approximation for dynamic SBNs, a neural network approximation and a form of mean field approximation (Saul and Jordan, 1999). We first show that the previous neural network model of (Henderson, 2003) can be viewed as a coarse approximation to inference with ISBNs. We then propose an incremental mean field method, which results in an improved approximation over the neural network but remains tractable. The resulting parser achieves significantly higher accuracy than the neural network parser (90.0% F-measure vs 89.1%). We argue that this correlation between better approximation and better accuracy suggests that dynamic SBNs are a good abstract model for natural language parsing.

2 Sigmoid Belief Networks

A belief network, or a Bayesian network, is a directed acyclic graph which encodes statistical dependencies between variables. Each variable S_i in the graph has an associated conditional probability distributions $P(S_i|Par(S_i))$ over its values given the values of its parents $Par(S_i)$ in the graph. A Sigmoid Belief Network (Neal, 1992) is a particular type of belief networks with binary variables and conditional probability distributions in the form of the logistic sigmoid function:

$$P(S_i{=}1|Par(S_i)) = \frac{1}{1{+}\exp(-\sum_{S_j \in Par(S_i)} J_{ij} S_j)},$$

where J_{ij} is the weight for the edge from variable S_j to variable S_i. In this paper we consider a generalized version of SBNs where we allow variables with any range of discrete values. We thus generalize the logistic sigmoid function to the normalized exponential (a.k.a. softmax) function to define the conditional probabilities for non-binary variables.

Exact inference with all but very small SBNs is not tractable. Initially sampling methods were used (Neal, 1992), but this is also not feasible for large networks, especially for the dynamic models of the type described in section 2.2. Variational methods have also been proposed for approximating SBNs (Saul and Jordan, 1999). The main idea of variational methods (Jordan et al., 1999) is, roughly, to construct a tractable approximate model with a number of free parameters. The free parameters are set so that the resulting approximate model is as close as possible to the original graphical model for a given inference problem.

2.1 Mean Field Approximation Methods

The simplest example of a variation method is the mean field method, originally introduced in statistical mechanics and later applied to unsupervised neural networks in (Hinton et al., 1995). Let us denote the set of visible variables in the model (i.e. the inputs and outputs) by V and hidden variables by $H = h_1, \ldots, h_l$. The mean field method uses a fully factorized distribution Q as the approximate model:

$$Q(H|V) = \prod_i Q_i(h_i|V).$$

where each Q_i is the distribution of an individual latent variable. The independence between the variables h_i in this approximate distribution Q does not imply independence of the free parameters which define the Q_i. These parameters are set to minimize the Kullback-Leibler divergence (Cover and Thomas, 1991) between the approximate distribution $Q(H|V)$ and the true distribution $P(H|V)$:

$$KL(Q\|P) = \sum_H Q(H|V) \ln \frac{Q(H|V)}{P(H|V)}, \qquad (1)$$

or, equivalently, to maximize the expression:

$$L_V = \sum_H Q(H|V) \ln \frac{P(H,V)}{Q(H|V)}. \qquad (2)$$

The expression L_V is a lower bound on the log-likelihood $\ln P(V)$. It is used in the mean field theory (Saul and Jordan, 1999) to approximate the likelihood. However, in our case of dynamic graphical models, we have to use a different approach which allows us to construct an incremental parsing method without needing to introduce the additional parameters proposed in (Saul and Jordan, 1999). We will describe our modification of the mean field method in section 3.3.

2.2 Dynamics

Dynamic Bayesian networks are Bayesian networks applied to arbitrarily long sequences. A new set of variables is instantiated for each position in the sequence, but the edges and weights for these variables are the same as in other positions. The edges which connect variables instantiated for different positions must be directed forward in the sequence, thereby allowing a temporal interpretation of the sequence. Typically a dynamic Bayesian Network will only involve edges between adjacent positions in the sequence (i.e. they are Markovian), but in our parsing models the pattern of interconnection is determined by structural locality, rather than sequence locality, as in the neural networks of (Henderson, 2003).

Using structural locality to define the graph in a dynamic SBN means that the subgraph of edges with destinations at a given position cannot be determined until all the parser decisions for previous positions have been chosen. We therefore call these models *Incremental* SBNs, because, at any given position in the parse, we only know the graph of edges for that position and previous positions in the parse. For example in figure 1, discussed below, it would not be possible to draw the portion of the graph after t, because we do not yet know the decision d_k^t.

The incremental specification of model structure means that we cannot use an undirected graphical model, such as Conditional Random Fields. With a directed dynamic model, all edges connecting the known portion of the graph to the unknown portion of the graph are directed toward the unknown portion. Also there are no variables in the unknown portion of the graph whose values are known (i.e. no visible variables), because at each step in a history-based model the decision probability is conditioned only on the parsing history. Only visible variables can result in information being reflected backward through a directed edge, so it is impossible for anything in the unknown portion of the graph to affect the probabilities in the known portion of the graph. Therefore inference can be performed by simply ignoring the unknown portion of the graph, and there is no need to sum over all possible structures for the unknown portion of the graph, as would be necessary for an undirected graphical model.

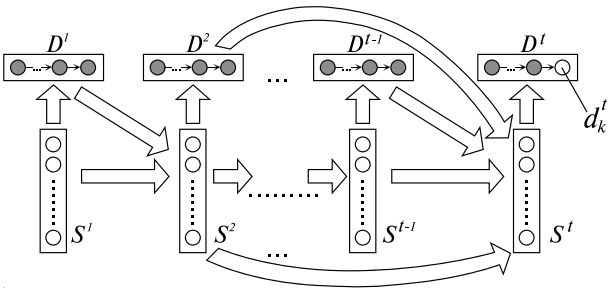


Figure 1: Illustration of an ISBN.

3 The Probabilistic Model of Parsing

In this section we present our framework for syntactic parsing with dynamic Sigmoid Belief Networks. We first specify the form of SBN we propose, namely ISBNs, and then two methods for approximating the inference problems required for parsing. We only consider generative models of parsing, since generative probability models are simpler and we are focused on probability estimation, not decision making. Although the most accurate parsing models (Charniak and Johnson, 2005; Henderson, 2004; Collins, 2000) are discriminative, all the most accurate discriminative models make use of a generative model. More accurate generative models should make the discriminative models which use them more accurate as well. Also, there are some applications, such as language modeling, which require generative models.

3.1 The Graphical Model

In ISBNs, we use a history-based model, which decomposes the probability of the parse as:

$$P(T) = P(D^1, ..., D^m) = \prod_t P(D^t | D^1, \ldots, D^{t-1}),$$

where T is the parse tree and $D^1, \ldots, D^m$ is its equivalent sequence of parser decisions. Instead of treating each D^t as atomic decisions, it is convenient to further split them into a sequence of elementary decisions $D^t = d_1^t, \ldots, d_n^t$:

$$P(D^t | D^1, \ldots, D^{t-1}) = \prod_k P(d_k^t | h(t,k)),$$

where $h(t,k)$ denotes the parsing history $D^1, \ldots, D^{t-1}, d_1^t, \ldots, d_{k-1}^t$. For example, a

decision to create a new constituent can be divided in two elementary decisions: deciding to create a constituent and deciding which label to assign to it. We use a graphical model to define our proposed class of probability models. An example graphical model for the computation of $P(d_k^t|h(t,k))$ is illustrated in figure 1.

The graphical model is organized into vectors of variables: latent state variable vectors $S^{t'} = s_1^{t'}, \ldots, s_n^{t'}$, representing an intermediate state of the parser at derivation step t', and decision variable vectors $D^{t'} = d_1^{t'}, \ldots, d_l^{t'}$, representing a parser decision at derivation step t', where $t' \leq t$. Variables whose value are given at the current decision (t,k) are shaded in figure 1, latent and output variables are left unshaded.

As illustrated by the arrows in figure 1, the probability of each state variable $s_i^{t'}$ depends on all the variables in a finite set of relevant previous state and decision vectors, but there are no *direct* dependencies between the different variables in a single state vector. Which previous state and decision vectors are connected to the current state vector is determined by a set of structural relations specified by the parser designer. For example, we could select the most recent state where the same constituent was on the top of the stack, and a decision variable representing the constituent's label. Each such selected relation has its own distinct weight matrix for the resulting edges in the graph, but the same weight matrix is used at each derivation position where the relation is relevant.

As indicated in figure 1, the probability of each elementary decision $d_k^{t'}$ depends both on the current state vector $S^{t'}$ and on the previously chosen elementary action $d_{k-1}^{t'}$ from $D^{t'}$. This probability distribution has the form of a normalized exponential:

$$P(d_k^{t'} = d|S^{t'}, d_{k-1}^{t'}) = \frac{\Phi_{h(t',k)}(d)\, e^{\sum_j W_{dj} s_j^{t'}}}{\sum_{d'} \Phi_{h(t',k)}(d')\, e^{\sum_j W_{d'j} s_j^{t'}}}, \quad (3)$$

where $\Phi_{h(t',k)}$ is the indicator function of a set of elementary decisions that may possibly follow the parsing history $h(t',k)$, and the W_{dj} are the weights.

For our experiments, we replicated the same pattern of interconnection between state variables as described in (Henderson, 2003).[1] We also used the same left-corner parsing strategy, and the same set of decisions, features, and states. We refer the reader to (Henderson, 2003) for details.

Exact computation with this model is not tractable. Sampling of parse trees from the model is not feasible, because a generative model defines a joint model of both a sentence and a tree, thereby requiring sampling over the space of sentences. Gibbs sampling (Geman and Geman, 1984) is also impossible, because of the huge space of variables and need to resample after making each new decision in the sequence. Thus, we know of no reasonable alternatives to the use of variational methods.

3.2 A Feed-Forward Approximation

The first model we consider is a strictly incremental computation of a variational approximation, which we will call the feed-forward approximation. It can be viewed as the simplest form of mean field approximation. As in any mean field approximation, each of the latent variables is independently distributed. But unlike the general case of mean field approximation, in the feed-forward approximation we only allow the parameters of the distributions Q_i to depend on the distributions of their parents. This additional constraint increases the potential for a large Kullback-Leibler divergence with the true model, defined in expression (1), but it significantly simplifies the computations.

The set of hidden variables H in our graphical model consists of all the state vectors $S^{t'}$, $t' \leq t$, and the last decision d_k^t. All the previously observed decisions $h(t,k)$ comprise the set of visible variables V. The approximate fully factorisable distribution $Q(H|V)$ can be written as:

$$Q(H|V) = q_k^t(d_k^t) \prod_{t',i} \left(\mu_i^{t'}\right)^{s_i^{t'}} \left(1 - \mu_i^{t'}\right)^{1 - s_i^{t'}}.$$

where $\mu_i^{t'}$ is the free parameter which determines the distribution of state variable i at position t', namely its mean, and $q_k^t(d_k^t)$ is the free parameter which determines the distribution over decisions d_k^t.

Because we are only allowed to use information about the distributions of the parent variables to

[1] In the neural network of (Henderson, 2003), our variables map to their "units", and our dependencies/edges map to their "links".

compute the free parameters $\mu_i^{t'}$, the optimal assignment of values to the $\mu_i^{t'}$ is:

$$\mu_i^{t'} = \sigma\left(\eta_i^{t'}\right),$$

where σ denotes the logistic sigmoid function and $\eta_i^{t'}$ is a weighted sum of the parent variables' means:

$$\eta_i^{t'} = \sum_{t'' \in RS(t')} \sum_j J_{ij}^{\tau(t',t'')} \mu_j^{t''} + \sum_{t'' \in RD(t')} \sum_k B_{id_k^{t''}}^{\tau(t',t'')}, \quad (4)$$

where $RS(t')$ is the set of previous positions with edges from their state vectors to the state vector at t', $RD(t')$ is the set of previous positions with edges from their decision vectors to the state vector at t', $\tau(t', t'')$ is the relevant relation between the position t'' and the position t', and J_{ij}^{τ} and B_{id}^{τ} are weight matrices.

In order to maximize (2), the approximate distribution of the next decisions $q_k^t(d)$ should be set to

$$q_k^t(d) = \frac{\Phi_{h(t,k)}(d)\, e^{\sum_j W_{dj}\mu_j^t}}{\sum_{d'} \Phi_{h(t,k)}(d')\, e^{\sum_j W_{d'j}\mu_j^t}}, \quad (5)$$

as follows from expression (3). The resulting estimate of the tree probability is given by:

$$P(T) \approx \prod_{t,k} q_k^t(d_k^t).$$

This approximation method replicates exactly the computation of the feed-forward neural network in (Henderson, 2003), where the above means $\mu_i^{t'}$ are equivalent to the neural network hidden unit activations. Thus, that neural network probability model can be regarded as a simple approximation to the graphical model introduced in section 3.1.

In addition to the drawbacks shared by any mean field approximation method, this feed-forward approximation cannot capture backward reasoning. By backward (a.k.a. top-down) reasoning we mean the need to update the state vector means $\mu_i^{t'}$ after observing a decision d_k^t, for $t' \leq t$. The next section discusses how backward reasoning can be incorporated in the approximate model.

3.3 A Mean Field Approximation

This section proposes a more accurate way to approximate ISBNs with mean field methods, which we will call the mean field approximation. Again, we are interested in finding the distribution Q which maximizes the quantity L_V in expression (2). The decision distribution $q_k^t(d_k^t)$ maximizes L_V when it has the same dependence on the state vector means μ_k^t as in the feed-forward approximation, namely expression (5). However, as we mentioned above, the feed-forward computation does not allow us to compute the optimal values of state means $\mu_i^{t'}$.

Optimally, after each new decision d_k^t, we should recompute all the means $\mu_i^{t'}$ for all the state vectors $S^{t'}$, $t' \leq t$. However, this would make the method intractable, due to the length of derivations in constituent parsing and the interdependence between these means. Instead, after making each decision d_k^t and adding it to the set of visible variables V, we recompute only means of the current state vector S^t.

The denominator of the normalized exponential function in (3) does not allow us to compute L_V exactly. Instead, we use a simple first order approximation:

$$\begin{aligned} &E_Q[\ln \sum_d \Phi_{h(t,k)}(d) \exp(\sum_j W_{dj} s_j^t)] \\ &\quad \approx \ln \sum_d \Phi_{h(t,k)}(d) \exp(\sum_j W_{dj}\mu_j^t), \end{aligned} \quad (6)$$

where the expectation $E_Q[\ldots]$ is taken over the state vector S^t distributed according to the approximate distribution Q.

Unfortunately, even with this assumption there is no analytic way to maximize L_V with respect to the means μ_k^t, so we need to use numerical methods. Assuming (6), we can rewrite the expression (2) as follows, substituting the true $P(H, V)$ defined by the graphical model and the approximate distribution $Q(H|V)$, omitting parts independent of μ_k^t:

$$\begin{aligned} L_V^{t,k} = &\sum_i -\mu_i^t \ln \mu_i^t - (1-\mu_i^t) \ln\left(1-\mu_i^t\right) \\ &+\mu_i^t \eta_i^t + \sum_{k'<k} \Phi_{h(t,k')}(d_{k'}^t) \sum_j W_{d_{k'}^t j} \mu_j^t \\ &- \sum_{k'<k} \ln\left(\sum_d \Phi_{h(t,k')}(d) \exp(\sum_j W_{dj}\mu_j^t)\right), \end{aligned} \quad (7)$$

here, η_i^t is computed from the previous relevant state means and decisions as in (4). This expression is

concave with respect to the parameters μ_i^t, so the global maximum can be found. We use coordinate-wise ascent, where each μ_i^t is selected by an efficient line search (Press et al., 1996), while keeping other $\mu_{i'}^t$ fixed.

3.4 Parameter Estimation

We train these models to maximize the fit of the *approximate* model to the data. We use gradient descent and a maximum likelihood objective function. This requires computation of the gradient of the approximate log-likelihood with respect to the model parameters. In order to compute these derivatives, the error should be propagated all the way back through the structure of the graphical model. For the feed-forward approximation, computation of the derivatives is straightforward, as in neural networks. But for the mean field approximation, it requires computation of the derivatives of the means μ_i^t with respect to the other parameters in expression (7). The use of a numerical search in the mean field approximation makes the analytical computation of these derivatives impossible, so a different method needs to be used to compute their values. If maximization of $L_V^{t,k}$ is done until convergence, then the derivatives of $L_V^{t,k}$ with respect to μ_i^t are close to zero:

$$F_i^{t,k} = \frac{\partial L_V^{t,k}}{\partial \mu_i^t} \approx 0 \text{ for all } i.$$

This system of equations allows us to use implicit differentiation to compute the needed derivatives.

4 Experimental Evaluation

In this section we evaluate the two approximations to dynamic SBNs discussed in the previous section, the feed-forward method equivalent to the neural network of (Henderson, 2003) (NN method) and the mean field method (MF method). The hypothesis we wish to test is that the more accurate approximation of dynamic SBNs will result in a more accurate model of constituent structure parsing. If this is true, then it suggests that dynamic SBNs of the form proposed here are a good abstract model of the nature of natural language parsing.

We used the Penn Treebank WSJ corpus (Marcus et al., 1993) to perform the empirical evaluation of the considered approaches. It is expensive to train the MF approximation on the whole WSJ corpus, so instead we use only sentences of length at most 15, as in (Taskar et al., 2004) and (Turian and Melamed, 2006). The standard split of the corpus into training (sections 2–22, 9,753 sentences), validation (section 24, 321 sentences), and testing (section 23, 603 sentences) was performed.[2]

	R	P	F_1
Bikel, 2004	87.9	88.8	88.3
Taskar et al., 2004	89.1	89.1	89.1
NN method	**89.1**	**89.2**	**89.1**
Turian and Melamed, 2006	89.3	89.6	89.4
MF method	**89.3**	**90.7**	**90.0**
Charniak, 2000	90.0	90.2	90.1

Table 1: Percentage labeled constituent recall (R), precision (P), combination of both (F_1) on the testing set.

As in (Henderson, 2003; Turian and Melamed, 2006) we used a publicly available tagger (Ratnaparkhi, 1996) to provide the part-of-speech tag for each word in the sentence. For each tag, there is an unknown-word vocabulary item which is used for all those words which are not sufficiently frequent with that tag to be included individually in the vocabulary. We only included a specific tag-word pair in the vocabulary if it occurred at least 20 time in the training set, which (with tag-unknown-word pairs) led to the very small vocabulary of 567 tag-word pairs.

During parsing with both the NN method and the MF method, we used beam search with a post-word beam of 10. Increasing the beam size beyond this value did not significantly effect parsing accuracy. For both of the models, the state vector size of 40 was used. All the parameters for both the NN and MF models were tuned on the validation set. A single best model of each type was then applied to the final testing set.

Table 1 lists the results of the NN approximation and the MF approximation, along with results of dif-

[2] Training of our MF method on this subset of WSJ took less than 6 days on a standard desktop PC. We would expect that a model for the entire WSJ corpus can be trained in about 3 months time. The training time is about linear with the number of words, but a larger state vector is needed to accommodate all the information. The long training times on the entire WSJ would not allow us to tune the model parameters properly, which would have increased the randomness of the empirical comparison, although it would be feasible for building a system.

ferent generative and discriminative parsing methods (Bikel, 2004; Taskar et al., 2004; Turian and Melamed, 2006; Charniak, 2000) evaluated in the same experimental setup. The MF model improves over the baseline NN approximation, with an error reduction in F-measure exceeding 8%. This improvement is statically significant.[3] The MF model achieves results which do not appear to be significantly different from the results of the best model in the list (Charniak, 2000). It should also be noted that the model (Charniak, 2000) is the most accurate generative model on the standard WSJ parsing benchmark, which confirms the viability of our generative model.

These experimental results suggest that Incremental Sigmoid Belief Networks are an appropriate model for natural language parsing. Even approximations such as those tested here, with a very strong factorisability assumption, allow us to build quite accurate parsing models. The main drawback of our proposed mean field approach is the relative computational complexity of the numerical procedure used to maximize $L_V^{t,k}$. But this approximation has succeeded in showing that a more accurate approximation of ISBNs results in a more accurate parser. We believe this provides strong justification for more accurate approximations of ISBNs for parsing.

5 Related Work

There has not been much previous work on graphical models for full parsing, although recently several latent variable models for parsing have been proposed (Koo and Collins, 2005; Matsuzaki et al., 2005; Riezler et al., 2002). In (Koo and Collins, 2005), an undirected graphical model is used for parse reranking. Dependency parsing with dynamic Bayesian networks was considered in (Peshkin and Savova, 2005), with limited success. Their model is very different from ours. Roughly, it considered the whole sentence at a time, with the graphical model being used to decide which words correspond to leaves of the tree. The chosen words are then removed from the sentence and the model is recursively applied to the reduced sentence.

Undirected graphical models, in particular Conditional Random Fields, are the standard tools for shallow parsing (Sha and Pereira, 2003). However, shallow parsing is effectively a sequence labeling problem and therefore differs significantly from full parsing. As discussed in section 2.2, undirected graphical models do not seem to be suitable for history-based full parsing models.

Sigmoid Belief Networks were used originally for character recognition tasks, but later a dynamic modification of this model was applied to the reinforcement learning task (Sallans, 2002). However, their graphical model, approximation method, and learning method differ significantly from those of this paper.

6 Conclusions

This paper proposes a new generative framework for constituent parsing based on dynamic Sigmoid Belief Networks with vectors of latent variables. Exact inference with the proposed graphical model (called Incremental Sigmoid Belief Networks) is not tractable, but two approximations are considered. First, it is shown that the neural network parser of (Henderson, 2003) can be considered as a simple feed-forward approximation to the graphical model. Second, a more accurate but still tractable approximation based on mean field theory is proposed. Both methods are empirically compared, and the mean field approach achieves significantly better results, which are non-significantly different from the results of the most accurate generative parsing model (Charniak, 2000) on our testing set. The fact that a more accurate approximation leads to a more accurate parser suggests that ISBNs are a good abstract model for constituent structure parsing. This empirical result motivates research into more accurate approximations of dynamic SBNs.

We focused in this paper on generative models of parsing. The results of such a generative model can be easily improved by a discriminative reranking model, even without any additional feature engineering. For example, the discriminative training techniques successfully applied in (Henderson, 2004) to the feed-forward neural network model can be directly applied to the mean field model proposed in this paper. The same is true for reranking with data-defined kernels, with which we would

[3] We measured signifi cance of all the experiments in this paper with the randomized signifi cance test (Yeh, 2000).

expect similar improvements as were achieved with the neural network parser (Henderson and Titov, 2005). Such improvements should situate the resulting model among the best current parsing models.

References

Dan M. Bikel. 2004. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4).

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fi ne n-best parsing and MaxEnt discriminative reranking. In *Proc. ACL*, pages 173–180, Ann Arbor, MI.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. ACL*, pages 132–139, Seattle, Washington.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proc. ICML*, pages 175–182, Stanford, CA.

Thomas M. Cover and Joy A. Thomas. 1991. *Elements of Information Theory*. John Wiley, New York, NY.

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

James Henderson and Ivan Titov. 2005. Data-defi ned kernels for parse reranking derived from probabilistic models. In *Proc. ACL*, Ann Arbor, MI.

James Henderson. 2003. Inducing history representations for broad coverage statistical parsing. In *Proc. HLT-NAACL*, pages 103–110, Edmonton, Canada.

James Henderson. 2004. Discriminative training of a neural network statistical parser. In *Proc. ACL*, Barcelona, Spain.

G. Hinton, P. Dayan, B. Frey, and R. Neal. 1995. The wake-sleep algorithm for unsupervised neural networks. *Science*, 268:1158–1161.

M. I. Jordan, Z.Ghahramani, T. S. Jaakkola, and L. K. Saul. 1999. An introduction to variational methods for graphical models. In Michael I. Jordan, editor, *Learning in Graphical Models*. MIT Press, Cambridge, MA.

Terry Koo and Michael Collins. 2005. Hidden-variable models for discriminative reranking. In *Proc. EMNLP*, Vancouver, B.C., Canada.

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. 2005. Probabilistic CFG with latent annotations. In *Proc. ACL*, Ann Arbor, MI.

Radford Neal. 1992. Connectionist learning of belief networks. *Artificial Intelligence*, 56:71–113.

Leon Peshkin and Virginia Savova. 2005. Dependency parsing with dynamic bayesian network. In *AAAI, 20th National Conference on Artificial Intelligence*, Pittsburgh, Pennsylvania.

W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. 1996. *Numerical Recipes*. Cambridge University Press, Cambridge, UK.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proc. EMNLP*, pages 133–142, Univ. of Pennsylvania, PA.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proc. ACL*, Philadelphia, PA.

Brian Sallans. 2002. *Reinforcement Learning for Factored Markov Decision Processes*. Ph.D. thesis, University of Toronto, Toronto, Canada.

Lawrence K. Saul and Michael I. Jordan. 1999. A mean fi eld learning algorithm for unsupervised neural networks. In Michael I. Jordan, editor, *Learning in Graphical Models*, pages 541–554. MIT Press, Cambridge, MA.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fi elds. In *Proc. HLT-NAACL*, Edmonton, Canada.

Ben Taskar, Dan Klein, Michael Collins, Daphne Koller, and Christopher Manning. 2004. Max-margin parsing. In *Proc. EMNLP*, Barcelona, Spain.

Joseph Turian and Dan Melamed. 2006. Advances in discriminative parsing. In *Proc. COLING-ACL*, Sydney, Australia.

Alexander Yeh. 2000. More accurate tests for the statistical signifi cance of the result differences. In *Proc. COLING*, pages 947–953, Saarbruken, Germany.

Corpus Effects on the Evaluation of Automated Transliteration Systems

Sarvnaz Karimi **Andrew Turpin** **Falk Scholer**
School of Computer Science and Information Technology
RMIT University, GPO Box 2476V, Melbourne 3001, Australia
{sarvnaz,aht,fscholer}@cs.rmit.edu.au

Abstract

Most current machine transliteration systems employ a corpus of known source-target word pairs to train their system, and typically evaluate their systems on a similar corpus. In this paper we explore the performance of transliteration systems on corpora that are varied in a controlled way. In particular, we control the number, and prior language knowledge of human transliterators used to construct the corpora, and the origin of the source words that make up the corpora. We find that the word accuracy of automated transliteration systems can vary by up to 30% (in absolute terms) depending on the corpus on which they are run. We conclude that at least four human transliterators should be used to construct corpora for evaluating automated transliteration systems; and that although absolute word accuracy metrics may not translate across corpora, the relative rankings of system performance remains stable across differing corpora.

1 Introduction

Machine transliteration is the process of transforming a word written in a source language into a word in a target language without the aid of a bilingual dictionary. Word pronunciation is preserved, as far as possible, but the script used to render the target word is different from that of the source language. Transliteration is applied to proper nouns and out-of-vocabulary terms as part of machine translation and cross-lingual information retrieval (CLIR) (AbdulJaleel and Larkey, 2003; Pirkola et al., 2006). Several transliteration methods are reported in the literature for a variety of languages, with their performance being evaluated on multilingual corpora. Source-target pairs are either extracted from bilingual documents or dictionaries (AbdulJaleel and Larkey, 2003; Bilac and Tanaka, 2005; Oh and Choi, 2006; Zelenko and Aone, 2006), or gathered explicitly from human transliterators (Al-Onaizan and Knight, 2002; Zelenko and Aone, 2006). Some evaluations of transliteration methods depend on a single unique transliteration for each source word, while others take multiple target words for a single source word into account. In their work on transliterating English to Persian, Karimi et al. (2006) observed that the content of the corpus used for evaluating systems could have dramatic affects on the reported accuracy of methods.

The effects of corpus composition on the evaluation of transliteration systems has not been specifically studied, with only implicit experiments or claims made in the literature such as introducing the effects of different transliteration models (AbdulJaleel and Larkey, 2003), language families (Lindén, 2005) or application based (CLIR) evaluation (Pirkola et al., 2006). In this paper, we report our experiments designed to explicitly examine the effect that varying the underlying corpus used in both training and testing systems has on transliteration accuracy. Specifically, we vary the number of human transliterators that are used to construct the corpus; and the origin of the English words used in the corpus.

Our experiments show that the word accuracy of automated transliteration systems can vary by up to 30% (in absolute terms), depending on the corpus used. Despite the wide range of absolute values

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 640–647,
Prague, Czech Republic, June 2007.

in performance, the ranking of our two transliteration systems was preserved on all corpora. We also find that a human's confidence in the language from which they are transliterating can affect the corpus in such a way that word accuracy rates are altered.

2 Background

Machine transliteration methods are divided into grapheme-based (AbdulJaleel and Larkey, 2003; Lindén, 2005), phoneme-based (Jung et al., 2000; Virga and Khudanpur, 2003) and combined techniques (Bilac and Tanaka, 2005; Oh and Choi, 2006). Grapheme-based methods derive transformation rules for character combinations in the source text from a training data set, while phoneme-based methods use an intermediate phonetic transformation. In this paper, we use two grapheme-based methods for English to Persian transliteration. During a training phase, both methods derive rules for transforming character combinations (*segments*) in the source language into character combinations in the target language with some probability.

During transliteration, the source word s_i is segmented and rules are chosen and applied to each segment according to heuristics. The probability of a resulting word is the product of the probabilities of the applied rules. The result is a list of target words sorted by their associated probabilities, L^i.

The first system we use (SYS-1) is an n-gram approach that uses the last character of the previous source segment to condition the choice of the rule for the current source segment. This system has been shown to outperform other n-gram based methods for English to Persian transliteration (Karimi et al., 2006).

The second system we employ (SYS-2) makes use of some explicit knowledge of our chosen language pair, English and Persian, and is also on the collapsed-vowel scheme presented by Karimi et al. (2006). In particular, it exploits the tendency for runs of English vowels to be collapsed into a single Persian character, or perhaps omitted from the Persian altogether. As such, segments are chosen based on surrounding consonants and vowels. The full details of this system are not important for this paper; here we focus on the performance evaluation of systems, not the systems themselves.

2.1 System Evaluation

In order to evaluate the list L^i of target words produced by a transliteration system for source word s_i, a test corpus is constructed. The test corpus consists of a source word, s_i, and a list of possible target words $\{t_{ij}\}$, where $1 \leq j \leq d_i$, the number of distinct target words for source word s_i. Associated with each t_{ij} is a count n_{ij} which is the number of human transliterators who transliterated s_i into t_{ij}.

Often the test corpus is a proportion of a larger corpus, the remainder of which has been used for training the system's rule base. In this work we adopt the standard ten-fold cross validation technique for all of our results, where 90% of a corpus is used for training and 10% for testing. The process is repeated ten times, and the mean result taken. Forthwith, we use the term corpus to refer to the single corpus from which both training and test sets are drawn in this fashion.

Once the corpus is decided upon, a metric to measure the system's accuracy is required. The appropriate metric depends on the scenario in which the transliteration system is to be used. For example, in a machine translation application where only one target word can be inserted in the text to represent a source word, it is important that the word at the top of the system generated list of target words (by definition the most probable) is one of the words generated by a human in the corpus. More formally, the first word generated for source word s_i, L^i_1, must be one of $t_{ij}, 1 \leq j \leq d_i$. It may even be desirable that this is the target word most commonly used for this source word; that is, $L^i_1 = t_{ij}$ such that $n_{ij} \geq n_{ik}$, for all $1 \leq k \leq d_i$. Alternately, in a CLIR application, all variants of a source word might be required. For example, if a user searches for an English term "Tom" in Persian documents, the search engine should try and locate documents that contain both "تام" (3 letters:ت-ا-م) and "تم"(2 letters: ت-م), two possible transliterations of "Tom" that would be generated by human transliterators. In this case, a metric that counts the number of t_{ij} that appear in the top d_i elements of the system generated list, L^i, might be appropriate.

In this paper we focus on the "Top-1" case, where it is important for the most probable target word generated by the system, L^i_1 to be either the most pop-

ular t_{ij} (labeled the *Majority*, with ties broken arbitrarily), or just one of the t_{ij}'s (labeled *Uniform* because all possible transliterations are equally rewarded). A third scheme (labeled *Weighted*) is also possible where the reward for t_{ij} appearing as L_1^i is $n_{ij}/\sum_{j=1}^{d_i} n_{ij}$; here, each target word is given a weight proportional to how often a human transliterator chose that target word. Due to space considerations, we focus on the first two variants only.

In general, there are two commonly used metrics for transliteration evaluation: word accuracy (WA) and character accuracy (CA) (Hall and Dowling, 1980). In all of our experiments, CA based metrics closely mirrored WA based metrics, and so conclusions drawn from the data would be the same whether WA metrics or CA metrics were used. Hence we only discuss and report WA based metrics in this paper.

For each source word in the test corpus of K words, word accuracy calculates the percentage of correctly transliterated terms. Hence for the majority case, where every source word in the corpus only has one target word, the word accuracy is defined as

$$MWA = |\{s_i | L_1^i = t_{i1}, 1 \leq i \leq K\}|/K,$$

and for the *Uniform* case, where every target variant is included with equal weight in the corpus, the word accuracy is defined as

$$UWA = |\{s_i | L_1^i \in \{t_{ij}\}, 1 \leq i \leq K, 1 \leq j \leq d_i\}|/K.$$

2.2 Human Evaluation

To evaluate the level of agreement between transliterators, we use an agreement measure based on Mun and Eye (2004).

For any source word s_i, there are d_i different transliterations made by the n_i human transliterators ($n_i = \sum_{j=1}^{d_i} n_{ij}$, where n_{ij} is the number of times source word s_i was transliterated into target word t_{ij}). When any two transliterators agree on the same target word, there are two agreements being made: transliterator one agrees with transliterator two, and vice versa. In general, therefore, the total number of agreements made on source word s_i is $\sum_{j=1}^{d_i} n_{ij}(n_{ij}-1)$. Hence the total number of actual agreements made on the entire corpus of K words is

$$A_{act} = \sum_{i=1}^{K} \sum_{j=1}^{d_i} n_{ij}(n_{ij}-1).$$

The total number of possible agreements (that is, when all human transliterators agree on a single target word for each source word), is

$$A_{poss} = \sum_{i=1}^{K} n_i(n_i-1).$$

The proportion of overall agreement is therefore

$$P_A = \frac{A_{act}}{A_{poss}}.$$

2.3 Corpora

Seven transliterators (T1, T2, …, T7: all native Persian speakers from Iran) were recruited to transliterate 1500 proper names that we provided. The names were taken from lists of names written in English on English Web sites. Five hundred of these names also appeared in lists of names on Arabic Web sites, and five hundred on Dutch name lists. The transliterators were not told of the origin of each word. The entire corpus, therefore, was easily separated into three sub-corpora of 500 words each based on the origin of each word. To distinguish these collections, we use E_7, A_7 and D_7 to denote the English, Arabic and Dutch sub-corpora, respectively. The whole 1500 word corpus is referred to as EDA_7.

Dutch and Arabic were chosen with an assumption that most Iranian Persian speakers have little knowledge of Dutch, while their familiarity with Arabic should be in the second rank after English. All of the participants held at least a Bachelors degree. Table 1 summarizes the information about the transliterators and their perception of the given task. Participants were asked to scale the difficulty of the transliteration of each sub-corpus, indicated as a scale from 1 (*hard*) to 3 (*easy*). Similarly, the participants' confidence in performing the task was rated from 1 (*no confidence*) to 3 (*quite confident*). The level of familiarity with second languages was also reported based on a scale of zero (*not familiar*) to 3 (*excellent knowledge*).

The information provided by participants confirms our assumption of transliterators knowledge of second languages: high familiarity with English, some knowledge of Arabic, and little or no prior knowledge of Dutch. Also, the majority of them found the transliteration of English terms of medium difficulty, Dutch was considered mostly hard, and Arabic as easy to medium.

Transliterator	Second Language Knowledge				Difficulty,Confidence		
	English	Dutch	Arabic	Other	English	Dutch	Arabic
1	2	0	1	-	1,1	1,2	2,3
2	2	0	2	-	2,2	2,3	3,3
3	2	0	1	-	2,2	1,2	2,2
4	2	0	1	-	2,2	2,1	3,3
5	2	0	2	Turkish	2,2	1,1	3,2
6	2	0	1	-	2,2	1,1	3,3
7	2	0	1	-	2,2	1,1	2,2

Table 1: Transliterator's language knowledge (0=not familiar to 3=excellent knowledge), perception of difficulty (1=hard to 3=easy) and confidence (1=no confidence to 3=quite confident) in creating the corpus.

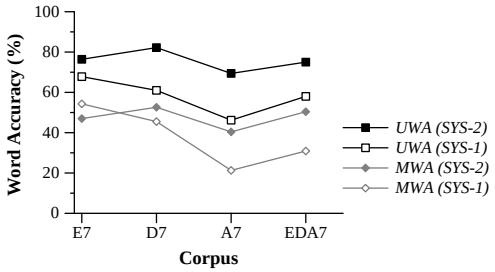


Figure 1: Comparison of the two evaluation metrics using the two systems on four corpora. (Lines were added for clarity, and do not represent data points.)

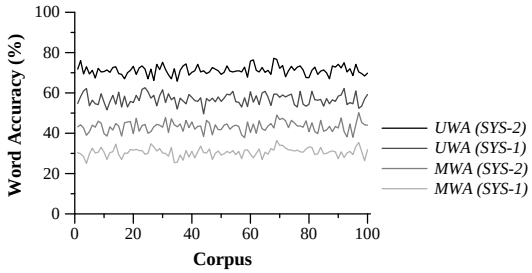


Figure 2: Comparison of the two evaluation metrics using the two systems on 100 randomly generated sub-corpora.

3 Results

Figure 1 shows the values of UWA and MWA for E_7, A_7, D_7 and EDA_7 using the two transliteration systems. Immediately obvious is that varying the corpora (x-axis) results in different values for word accuracy, whether by the *UWA* or *MWA* method. For example, if you chose to evaluate SYS-2 with the *UWA* metric on the D_7 corpus, you would obtain a result of 82%, but if you chose to evaluate it with the A_7 corpus you would receive a result of only 73%. This makes comparing systems that report results obtained on different corpora very difficult. Encouragingly, however, SYS-2 consistently outperforms the SYS-1 on all corpora for both metrics except *MWA* on $E7$. This implies that ranking system performance on the same corpus most likely yields a system ranking that is transferable to other corpora. To further investigate this, we randomly extracted 100 corpora of 500 word pairs from EDA_7 and ran the two systems on them and evaluated the results using both *MWA* and *UWA*. Both of the measures ranked the systems consistently using all these corpora (Figure 2).

As expected, the *UWA* metric is consistently higher than the *MWA* metric; it allows for the top transliteration to appear in any of the possible variants for that word in the corpus, unlike the *MWA* metric which insists upon a single target word. For example, for the E_7 corpus using the SYS-2 approach, *UWA* is 76.4% and *MWA* is 47.0%.

Each of the three sub-corpora can be further divided based on the seven individual transliterators, in different combinations. That is, construct a sub-corpus from T1's transliterations, T2's, and so on; then take all combinations of two transliterators, then three, and so on. In general we can construct ${}^7C_r$ such corpora from r transliterators in this fashion, all of which have 500 source words, but may have between one to seven different transliterations for each of those words.

Figure 3 shows the *MWA* for these sub-corpora. The x-axis shows the number of transliterators used to form the sub-corpora. For example, when $x = 3$, the performance figures plotted are achieved on corpora when taking all triples of the seven transliterator's transliterations.

From the boxplots it can be seen that performance varies considerably when the number of transliterators used to determine a majority vote is varied.

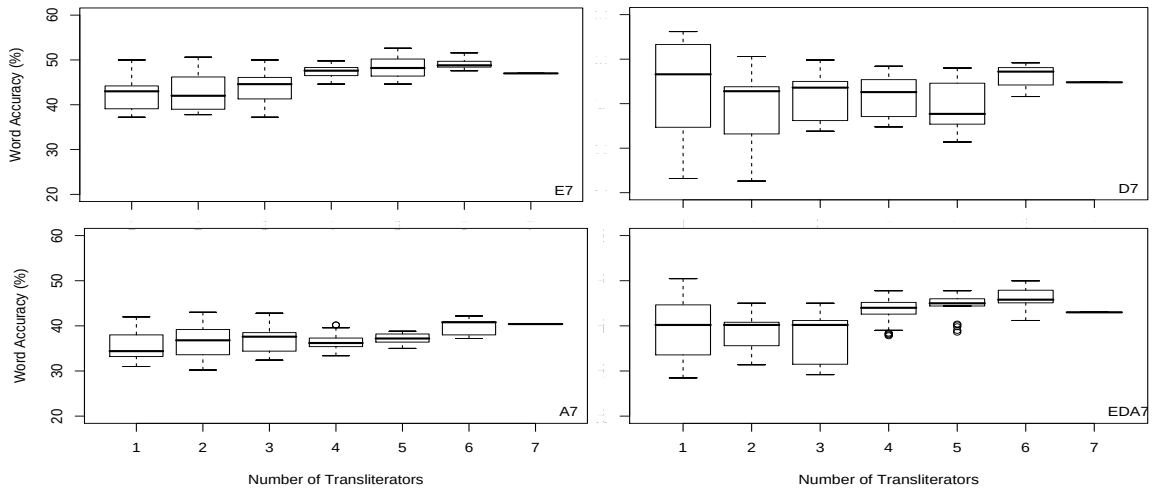


Figure 3: Performance on sub-corpora derived by combining the number of transliterators shown on the x-axis. Boxes show the 25th and 75th percentile of the *MWA* for all ${}^7C_x$ combinations of transliterators using SYS-2, with whiskers showing extreme values.

However, the changes do not follow a fixed trend across the languages. For E_7, the range of accuracies achieved is high when only two or three transliterators are involved, ranging from 37.0% to 50.6% in SYS-2 method and from 33.8% to 48.0% in SYS-1 (not shown) when only two transliterators' data are available. When more than three transliterators are used, the range of performance is noticeably smaller. Hence if at least four transliterators are used, then it is more likely that a system's *MWA* will be stable. This finding is supported by Papineni et al. (2002) who recommend that four people should be used for collecting judgments for machine translation experiments.

The corpora derived from A_7 show consistent median increases as the number of transliterators increases, but the median accuracy is lower than for other languages. The D_7 collection does not show any stable results until at least six transliterator's are used.

The results indicate that creating a collection used for the evaluation of transliteration systems, based on a "gold standard" created by only one human transliterator may lead to word accuracy results that could show a 10% absolute difference compared to results on a corpus derived using a different transliterator. This is evidenced by the leftmost box in each panel of the figure which has a wide range of results.

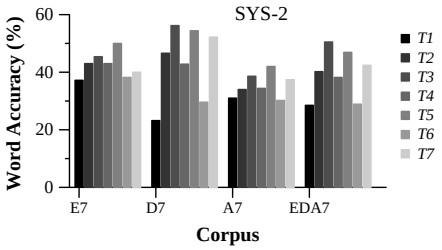


Figure 4: Word accuracy on the sub-corpora using only a single transliterator's transliterations.

Figure 4 shows this box in more detail for each collection, plotting the word accuracy for each user for all sub-corpora for SYS-2. The accuracy achieved varies significantly between transliterators; for example, for E_7 collections, word accuracy varies from 37.2% for T1 to 50.0% for T5. This variance is more obvious for the D_7 dataset where the difference ranges from 23.2% for $T1$ to 56.2% for $T3$. Origin language also has an effect: accuracy for the Arabic collection (A_7) is generally less than that of English (E_7). The Dutch collection (D_7), shows an unstable trend across transliterators. In other words, accuracy differs in a narrower range for Arabic and English, but in wider range for Dutch.

This is likely due to the fact that most transliterators found Dutch a difficult language to work with, as reported in Table 1.

3.1 Transliterator Consistency

To investigate the effect of invididual transliterator consistency on system accuracy, we consider the number of Persian characters used by each transliterator on each sub-corpus, and the average number of rules generated by SYS-2 on the ten training sets derived in the ten-fold cross validation process, which are shown in Table 2. For example, when transliterating words from E_7 into Persian, T3 only ever used 21 out of 32 characters available in the Persian alphabet; T7, on the other hand, used 24 different Persian characters. It is expected that an increase in number of characters or rules provides more "noise" for the automated system, hence may lead to lower accuracy. Superficially the opposite seems true for rules: the mean number of rules generated by SYS-2 is much higher for the EDA_7 corpus than for the A_7 corpus, and yet Figure 1 shows that word accuracy is higher on the EDA_7 corpus. A correlation test, however, reveals that there is no significant relationship between either the number of characters used, nor the number of rules generated, and the resulting word accuracy of SYS-2 (Spearman correlation, $p = 0.09$ (characters) and $p = 0.98$ (rules)).

A better indication of "noise" in the corpus may be given by the consistency with which a transliterator applies a certain rule. For example, a large number of rules generated from a particular transliterator's corpus may not be problematic if many of the rules get applied with a low probability. If, on the other hand, there were many rules with approximately equal probabilities, the system may have difficulty distinguishing when to apply some rules, and not others. One way to quantify this effect is to compute the *self entropy* of the rule distribution for each segment in the corpus for an individual. If p_{ij} is the probability of applying rule $1 \leq j \leq m$ when confronted with source segment i, then $H_i = -\sum_{j=1}^{m} p_{ij} \log_2 p_{ij}$ is the entropy of the probability distribution for that rule. H is maximized when the probabilities p_{ij} are all equal, and minimized when the probabilities are very skewed (Shannon, 1948). As an example, consider the rules: $t \rightarrow <$ت,0.5$>$, $t \rightarrow <$ط,0.3$>$ and $t \rightarrow <$د,0.2$>$; for which $H_t = 0.79$.

The expected entropy can be used to obtain a single entropy value over the whole corpus,

$$E = -\sum_{i=1}^{R} \frac{f_i}{S} H_i,$$

where H_i is the entropy of the rule probabilities for segment i, R is the total number of segments, f_i is the frequency with which segment i occurs at any position in all source words in the corpus, and S is the sum of all f_i.

The expected entropy for each transliterator is shown in Figure 5, separated by corpus. Comparison of this graph with Figure 4 shows that generally transliterators that have used rules inconsistently generate a corpus that leads to low accuracy for the systems. For example, T1 who has the lowest accuracy for all the collections in both methods, also has the highest expected entropy of rules for all the collections. For the E_7 collection, the maximum accuracy of 50.0%, belongs to $T5$ who has the minimum expected entropy. The same applies to the D_7 collection, where the maximum accuracy of 56.2% and the minimum expected entropy both belong to $T3$. These observations are confirmed by a statistically significant Spearman correlation between expected rule entropy and word accuracy ($r = -0.54, p = 0.003$). Therefore, the consistency with which transliterators employ their own internal rules in developing a corpus has a direct effect on system performance measures.

3.2 Inter-Transliterator Agreement and Perceived Difficulty

Here we present various agreement proportions (P_A from Section 2.2), which give a measure of consistency in the corpora across all users, as opposed to the entropy measure which gives a consistency measure for a single user. For E_7, P_A was 33.6%, for A_7 it was 33.3% and for D_7, agreement was 15.5%. In general, humans agree less than 33% of the time when transliterating English to Persian.

In addition, we examined agreement among transliterators based on their perception of the task difficulty shown in Table 1. For A_7, agreement among those who found the task *easy* was higher (22.3%) than those who found it in *medium* level

	E_7		D_7		A_7		EDA_7	
	Char	Rules	Char	Rules	Char	Rules	Char	Rules
T1	23	523	23	623	28	330	31	1075
T2	22	487	25	550	29	304	32	956
T3	21	466	20	500	28	280	31	870
T4	23	497	22	524	28	307	30	956
T5	21	492	22	508	28	296	29	896
T6	24	493	21	563	25	313	29	968
T7	24	495	21	529	28	299	30	952
Mean	23	493	22	542	28	304	30	953

Table 2: Number of characters used and rules generated using SYS-2, per transliterator.

(18.8%). P_A is 12.0% for those who found the D_7 collection *hard* to transliterate; while the six transliterators who found the E_7 collection difficulty *medium* had $P_A = 30.2\%$. Hence, the harder participants rated the transliteration task, the lower the agreement scores tend to be for the derived corpus.

Finally, in Table 3 we show word accuracy results for the two systems on corpora derived from transliterators grouped by perceived level of difficulty on A_7. It is readily apparent that SYS-2 outperforms SYS-1 on the corpus comprised of human transliterations from people who saw the task as easy with both word accuracy metrics; the relative improvement of over 50% is statistically significant (paired t-test on ten-fold cross validation runs). However, on the corpus composed of transliterations that were perceived as more difficult, "Medium", the advantage of SYS-2 is significantly eroded, but is still statistically significant for *UWA*. Here again, using only one transliteration, *MWA*, did not distinguish the performance of each system.

4 Discussion

We have evaluated two English to Persian transliteration systems on a variety of controlled corpora using evaluation metrics that appear in previous transliteration studies. Varying the evaluation corpus in a controlled fashion has revealed several interesting facts.

We report that human agreement on the English to Persian transliteration task is about 33%. The effect that this level of disagreement on the evaluation of systems has, can be seen in Figure 4, where word accuracy is computed on corpora derived from single transliterators. Accuracy can vary by up to 30% in absolute terms depending on the transliterator chosen. To our knowledge, this is the first paper

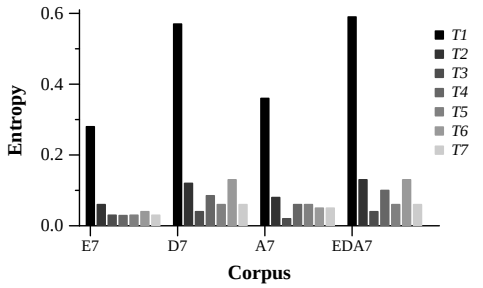


Figure 5: Entropy of the generated segments based on the collections created by different transliterators.

to report human agreement, and examine its effects on transliteration accuracy.

In order to alleviate some of these effects on the stability of word accuracy measures across corpora, we recommend that at least four transliterators are used to construct a corpus. Figure 3 shows that constructing a corpus with four or more transliterators, the range of possible word accuracies achieved is less than that of using fewer transliterators.

Some past studies do not use more than a single target word for every source word in the corpus (Bilac and Tanaka, 2005; Oh and Choi, 2006). Our results indicate that it is unlikely that these results would translate onto a corpus other than the one used in these studies, except in rare cases where human transliterators are in 100% agreement for a given language pair.

Given the nature of the English language, an English corpus can contain English words from a variety of different origins. In this study we have used English words from an Arabic and Dutch origin to show that word accuracy of the systems can vary by up to 25% (in absolute terms) depending on the origin of English words in the corpus, as demonstrated in Figure 1.

In addition to computing agreement, we also in-

	Perception	SYS-1	SYS-2	Relative Improvement (%)	
UWA	Easy	33.4	55.4	54.4	($p < 0.001$)
	Medium	44.6	48.4	8.52	($p < 0.001$)
MWA	Easy	23.2	36.2	56.0	($p < 0.001$)
	Medium	30.6	37.4	22.2	($p = 0.038$)

Table 3: System performance when A_7 is split into sub-corpora based on transliterators perception of the task (Easy or Medium).

vestigated the transliterator's perception of difficulty of the transliteration task with the ensuing word accuracy of the systems. Interestingly, when using corpora built from transliterators that perceive the task to be easy, there is a large difference in the word accuracy between the two systems, but on corpora built from transliterators who perceive the task to be more difficult, the gap between the systems narrows. Hence, a corpus applied for evaluation of transliteration should either be made carefully with transliterators with a variety of backgrounds, or should be large enough and be gathered from various sources so as to simulate different expectations of its expected non-homogeneous users.

The self entropy of rule probability distributions derived by the automated transliteration system can be used to measure the consistency with which individual transliterators apply their own rules in constructing a corpus. It was demonstrated that when systems are evaluated on corpora built by transliterators who are less consistent in their application of transliteration rules, word accuracy is reduced.

Given the large variations in system accuracy that are demonstrated by the varying corpora used in this study, we recommend that extreme care be taken when constructing corpora for evaluating transliteration systems. Studies should also give details of their corpora that would allow any of the effects observed in this paper to be taken into account.

Acknowledgments

This work was supported in part by the Australian government IPRS program (SK).

References

Nasreen AbdulJaleel and Leah S. Larkey. 2003. Statistical transliteration for English-Arabic cross-language information retrieval. In *Conference on Information and Knowledge Management*, pages 139–146.

Yaser Al-Onaizan and Kevin Knight. 2002. Machine transliteration of names in Arabic text. In *Proceedings of the ACL-02 workshop on Computational approaches to semitic languages*, pages 1–13.

Slaven Bilac and Hozumi Tanaka. 2005. Direct combination of spelling and pronunciation information for robust back-transliteration. In *Conference on Computational Linguistics and Intelligent Text Processing*, pages 413–424.

Patrick A. V. Hall and Geoff R. Dowling. 1980. Approximate string matching. *ACM Computing Survey*, 12(4):381–402.

Sung Young Jung, Sung Lim Hong, and Eunok Paek. 2000. An English to Korean transliteration model of extended Markov window. In *Conference on Computational Linguistics*, pages 383–389.

Sarvnaz Karimi, Andrew Turpin, and Falk Scholer. 2006. English to Persian transliteration. In *String Processing and Information Retrieval*, pages 255–266.

Krister Lindén. 2005. Multilingual modeling of cross-lingual spelling variants. *Information Retrieval*, 9(3):295–310.

Eun Young Mun and Alexander Von Eye, 2004. *Analyzing Rater Agreement: Manifest Variable Methods*. Lawrence Erlbaum Associates.

Jong-Hoon Oh and Key-Sun Choi. 2006. An ensemble of transliteration models for information retrieval. *Information Processing Management*, 42(4):980–1002.

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

Ari Pirkola, Jarmo Toivonen, Heikki Keskustalo, and Kalervo Järvelin. 2006. FITE-TRT: a high quality translation technique for OOV words. In *Proceedings of the 2006 ACM Symposium on Applied Computing*, pages 1043–1049.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423.

Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of proper names in cross-language applications. In *ACM SIGIR Conference on Research and Development on Information Retrieval*, pages 365–366.

Dmitry Zelenko and Chinatsu Aone. 2006. Discriminative methods for transliteration. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 612–617.

Collapsed Consonant and Vowel Models: New Approaches for English-Persian Transliteration and Back-Transliteration

Sarvnaz Karimi **Falk Scholer** **Andrew Turpin**
School of Computer Science and Information Technology
RMIT University, GPO Box 2476V, Melbourne 3001, Australia
{sarvnaz,fscholer,aht}@cs.rmit.edu.au

Abstract

We propose a novel algorithm for English to Persian transliteration. Previous methods proposed for this language pair apply a word alignment tool for training. By contrast, we introduce an alignment algorithm particularly designed for transliteration. Our new model improves the English to Persian transliteration accuracy by 14% over an n-gram baseline. We also propose a novel back-transliteration method for this language pair, a previously unstudied problem. Experimental results demonstrate that our algorithm leads to an absolute improvement of 25% over standard transliteration approaches.

1 Introduction

Translation of a text from a source language to a target language requires dealing with technical terms and proper names. These occur in almost any text, but rarely appear in bilingual dictionaries. The solution is the transliteration of such *out-of-dictionary* terms: a word from the source language is transformed to a word in the target language, preserving its pronunciation. Recovering the original word from the transliterated target is called back-transliteration. Automatic transliteration is important for many different applications, including machine translation, cross-lingual information retrieval and cross-lingual question answering.

Transliteration methods can be categorized into grapheme-based (AbdulJaleel and Larkey, 2003; Li et al., 2004), phoneme-based (Knight and Graehl, 1998; Jung et al., 2000), and combined (Bilac and Tanaka, 2005) approaches. Grapheme-based methods perform a direct orthographical mapping between source and target words, while phoneme-based approaches use an intermediate phonetic representation. Both grapheme- or phoneme-based methods usually begin by breaking the source word into segments, and then use a source segment to target segment mapping to generate the target word. The rules of this mapping are obtained by aligning already available transliterated word pairs (training data); alternatively, such rules can be handcrafted. From this perspective, past work is roughly divided into those methods which apply a word alignment tool such as GIZA++ (Och and Ney, 2003), and approaches that combine the alignment step into their main transliteration process.

Transliteration is language dependent, and methods that are effective for one language pair may not work as well for another. In this paper, we investigate the English-Persian transliteration problem. Persian (Farsi) is an Indo-European language, written in Arabic script from right to left, but with an extended alphabet and different pronunciation from Arabic. Our previous approach to English-Persian transliteration introduced the grapheme-based *collapsed-vowel* method, employing GIZA++ for source to target alignment (Karimi et al., 2006). We propose a new transliteration approach that extends the *collapsed-vowel* method. To meet Persian language transliteration requirements, we also propose a novel alignment algorithm in our training stage, which makes use of statistical information of

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 648–655,
Prague, Czech Republic, June 2007.

the corpus, transliteration specifications, and simple language properties. This approach handles possible consequences of elision (omission of sounds to make the word easier to read) and epenthesis (adding extra sounds to a word to make it fluent) in written target words that happen due to the change of language. Our method shows an absolute accuracy improvement of 14.2% over an n-gram baseline.

In addition, we investigate the problem of back-transliteration from Persian to English. To our knowledge, this is the first report of such a study. There are two challenges in Persian to English transliteration that makes it particularly difficult. First, written Persian omits short vowels, while only long vowels appear in texts. Second, monophthongization (changing diphthongs to monophthongs) is popular among Persian speakers when adapting foreign words into their language. To take these into account, we propose a novel method to form transformation rules by changing the normal segmentation algorithm. We find that this method significantly improves the Persian to English transliteration effectiveness, demonstrating an absolute performance gain of 25.1% over standard transliteration approaches.

2 Background

In general, transliteration consists of a training stage (running on a bilingual training corpus), and a generation – also called testing – stage.

The training step of a transliteration develops transformation rules mapping characters in the source to characters in the target language using knowledge of corresponding characters in transliterated pairs provided by an alignment. For example, for the source-target word pair (pat,پات), an alignment may map "p" to "پ" and "a" to "ا", and the training stage may develop the rule $pa \rightarrow$ا, with "ا" as the transliteration of "a" in the context of "pa". The generation stage applies these rules on a segmented source word, transforming it to a word in the target language.

Previous work on transliteration either employs a word alignment tool (usually GIZA++), or develops specific alignment strategies. Transliteration methods that use GIZA++ as their word pair aligner (AbdulJaleel and Larkey, 2003; Virga and Khudanpur, 2003; Karimi et al., 2006) have based their work on the assumption that the provided alignments are reliable. Gao et al. (2004) argue that precise alignment can improve transliteration effectiveness, experimenting on English-Chinese data and comparing IBM models (Brown et al., 1993) with phoneme-based alignments using direct probabilities.

Other transliteration systems focus on alignment for transliteration, for example the joint source-channel model suggested by Li et al. (2004). Their method outperforms the noisy channel model in direct orthographical mapping for English-Chinese transliteration. Li et al. also find that grapheme-based methods that use the joint source-channel model are more effective than phoneme-based methods due to removing the intermediate phonetic transformation step. Alignment has also been investigated for transliteration by adopting Covington's algorithm on cognate identification (Covington, 1996); this is a character alignment algorithm based on matching or skipping of characters, with a manually assigned cost of association. Covington considers consonant to consonant and vowel to vowel correspondence more valid than consonant to vowel. Kang and Choi (2000) revise this method for transliteration where a skip is defined as inserting a null in the target string when two characters do not match based on their phonetic similarities or their consonant and vowel nature. Oh and Choi (2002) revise this method by introducing *binding*, in which many to many correspondences are allowed. However, all of these approaches rely on the manually assigned penalties that need to be defined for each possible matching.

In addition, some recent studies investigate discriminative transliteration methods (Klementiev and Roth, 2006; Zelenko and Aone, 2006) in which each segment of the source can be aligned to each segment of the target, where some restrictive conditions based on the distance of the segments and phonetic similarities are applied.

3 The Proposed Alignment Approach

We propose an alignment method based on segment occurrence frequencies, thereby avoiding predefined matching patterns and penalty assignments. We also apply the observed tendency of aligning consonants

to consonants, and vowels to vowels, as a substitute for phonetic similarities. Many to many, one to many, one to null and many to one alignments can be generated.

3.1 Formulation

Our alignment approach consists of two steps: the first is based on the consonant and vowel nature of the word's letters, while the second uses a frequency-based sequential search.

Definition 1 *A bilingual corpus $\mathcal{B}$ is the set $\{(S,T)\}$, where $S = s_1..s_\ell$, $T = t_1..t_m$, s_i is a letter in the source language alphabet, and t_j is a letter in the target language alphabet.*

Definition 2 *Given some word, w, the consonant-vowel sequence $p = (C|V)^+$ for w is obtained by replacing each consonant with C and each vowel with V.*

Definition 3 *Given some consonant-vowel sequence, p, a reduced consonant-vowel sequence q replaces all runs of C's with $\mathcal{C}$, and all runs of V's with $\mathcal{V}$; hence $q = q'|q''$, $q' = \mathcal{V}(\mathcal{CV})^*(\mathcal{C}|\epsilon)$ and $q'' = \mathcal{C}(\mathcal{VC})^*(\mathcal{V}|\epsilon)$.*

For each natural language word, we can determine the *consonant-vowel* sequence (p) from which the *reduced consonant-vowel* sequence (q) can be derived, giving a common notation between two different languages, no matter which script either of them use. To simplify, semi-vowels and approximants (sounds intermediate between consonants and vowels, such as "w" and "y" in English) are treated according to their target language counterparts.

In general, for all the word pairs (S,T) in a corpus $\mathcal{B}$, an alignment can be achieved using the function

$$f : \mathcal{B} \rightarrow \mathcal{A}; (S,T) \mapsto (\hat{S}, \hat{T}, r).$$

The function f maps the word pair $(S,T) \in \mathcal{B}$ to the triple $(\hat{S}, \hat{T}, r) \in \mathcal{A}$ where $\hat{S}$ and $\hat{T}$ are substrings of S and T respectively. The frequency of this correspondence is denoted by r. $\mathcal{A}$ represents a set of substring alignments, and we use a per word alignment notation of a_{e2p} when aligning English to Persian and a_{p2e} for Persian to English.

3.2 Algorithm Details

Our algorithm consists of two steps.

Step 1 **(Consonant-Vowel based)**

For any word pair $(S,T) \in \mathcal{B}$, the corresponding *reduced consonant-vowel* sequences, q_S and q_T, are generated. If the sequences match, then the aligned consonant clusters and vowel sequences are added to the alignment set $\mathcal{A}$. If q_S does not match with q_T, the word pair remains unaligned in *Step 1*.

The assumption in this step is that transliteration of each vowel sequence of the source is a vowel sequence in the target language, and similarly for consonants. However, consonants do not always map to consonants, or vowels to vowels (for example, the English letter "s" may be written as "اس" in Persian which consists of one vowel and one consonant). Alternatively, they might be omitted altogether, which can be specified as the null string, ε. We therefore require a second step.

Step 2 **(Frequency based)**

For most natural languages, the maximum length of corresponding phonemes of each grapheme is a digraph (two letters) or at most a trigraph. Hence, alignment can be defined as a search problem that seeks for units with a maximum length of two or three in both strings that need to be aligned. In our approach, we search based on statistical occurrence data available from *Step 1*.

In *Step 2*, only those words that remain unaligned at the end of *Step 1* need to be considered. For each pair of words (S,T), matching proceeds from left to right, examining one of the three possible options of transliteration: single letter to single letter, digraph to single letter and single letter to digraph. Trigraphs are unnecessary in alignment as they can be effectively captured during transliteration generation, as we explain below.

We define four different valid alignments for the source ($S = s_1 s_2 \ldots s_i \ldots s_l$) and target ($T = t_1 t_2 \ldots t_j \ldots t_m$) strings: (s_i, t_j, r), $(s_i s_{i+1}, t_j, r)$, $(s_i, t_j t_{j+1}, r)$ and (s_i, ε, r). These four options are considered as the only possible valid alignments, and the most frequently occurring alignment (highest r) is chosen. These frequencies are dynamically updated after successfully aligning a pair. For exceptional situations, where there is no character in the target string to match with the source character s_i, it is aligned with the empty string.

It is possible that none of the four valid alignment

options have occurred previously (that is, $r = 0$ for each). This situation can arise in two ways: first, such a tuple may simply not have occurred in the training data; and, second, the previous alignment in the current string pair may have been incorrect. To account for this second possibility, a partial backtracking is considered. Most misalignments are derived from the simultaneous comparison of alignment possibilities, giving the highest priority to the most frequent. For example if S=bbc, T=بس and $\mathcal{A} = \{$(b,ب,100),(bb,ب,40),(c,س,60)$\}$, starting from the initial position s_1 and t_1, the first alignment choice is (b,ب,101). However immediately after, we face the problem of aligning the second "b". There are two solutions: inserting ε and adding the triple (b,ε,1), or backtracking the previous alignment and substituting that with the less frequent but possible alignment of (bb,ب,41). The second solution is a better choice as it adds less ambiguous alignments containing ε. At the end, the alignment set is updated as $\mathcal{A} = \{$(b,ب,100),(bb,ب,41),(c,س,61)$\}$.

In case of equal frequencies, we check possible subsequent alignments to decide on which alignment should be chosen. For example, if (b,ب,100) and (bb,ب,100) both exist as possible options, we consider if choosing the former leads to a subsequent ε insertion. If so, we opt for the latter.

At the end of a string, if just one character in the target string remains unaligned while the last alignment is a ε insertion, that final alignment will be substituted for ε. This usually happens when the alignment of final characters is not yet registered in the alignment set, mainly because Persian speakers tend to transliterate the final vowels to consonants to preserve their existence in the word. For example, in the word "Jose" the final "e" might be transliterated to "ه" which is a consonant ("h") and therefore is not captured in *Step 1*.

Backparsing

The process of aligning words explained above can handle words with already known components in the alignment set $\mathcal{A}$ (the frequency of occurrence is greater than zero). However, when this is not the case, the system may repeatedly insert ε while part or all of the target characters are left intact (unsuccessful alignment). In such cases, processing the source and target backwards helps to find the problematic substrings: *backparsing*.

The poorly aligned substrings of the source and target are taken as new pairs of strings, which are then reintroduced into the system as new entries. Note that they themselves are not subject to backparsing. Most strings of repeating nulls can be broken up this way, and in the worst case will remain as one tuple in the alignment set.

To clarify, consider the example given in Figure 1. For the word pair (patricia,ایشیرتپ), where an association between "c" and "ش" is not yet registered. Forward parsing, as shown in the figure, does not resolve all target characters; after the incorrect alignment of "c" with "ε", subsequent characters are also aligned with null, and the substring "ایش" remains intact. Backward parsing, shown in the next line of the figure, is also not successful. It is able to correctly align the last two characters of the string, before generating repeated null alignments. Therefore, the central region — substrings of the source and target which remained unaligned plus one extra aligned segment to the left and right — is entered as a new pair to the system (ici,یشی), as shown in the line labelled *Input 2* in the figure. This new input meets *Step 1* requirements, and is aligned successfully. The resulting tuples are then merged with the alignment set $\mathcal{A}$.

An advantage of our backparsing strategy is that it takes care of casual transliterations happening due to elision and epenthesis (adding or removing extra sounds). It is not only in translation that people may add extra words to make fluent target text; for transliteration also, it is possible that spurious characters are introduced for fluency. However, this often follows patterns, such as adding vowels to the target form. These irregularities are consistently covered in the backparsing strategy, where they remain connected to their previous character.

4 Transliteration Method

Transliteration algorithms use aligned data (the output from the alignment process, a_{e2p} or a_{p2e} alignment tuples) for training to derive transformation rules. These rules are then used to generate a target word T given a new input source word S.

Initial alignment set:	
$\mathcal{A}$ = {(p,پ,42),(a,ا,320),(a,ε,99),(a,ا‌ی,10),(a,ی,35),(r,ر,200),(i,ی,60),(i,ε,5),(c,س,80),(c,چ,25),(t,ت,51)}	
Input:	(patricia,پ‌ت‌ر‌ی‌ش‌ا) $q_S = \mathcal{CVCVCVC}$ $q_T = \mathcal{CVCVC}$
Step 1:	$q_S \neq q_T$
Forward alignment:	(p,پ,43), (a,ε,100), (t,ت,52), (r,ر,201), (i,ی,61), (c,ε,1), (i,ε,6), (a,ε,100)
Backward alignment:	(a,ا,321), (i,ی,61), (c,ε,1), (i,ε,6), (r,ε,1), (t,ε,1), (a,ε,100), (p,ε,1)
Input 2:	(ici,ی‌ش‌ی) $q_S = \mathcal{VCV}$ $q_T = \mathcal{VCV}$
Step 1:	(i,ی,61),(c,ش,1), (i,ی,61)
Final Alignment:	a_{e2p} = ((p,پ),(a,ε),(t,ت),((r,ر),(i,ی),(c,ش),(i,ی),(a,ا))
Updated alignment set:	
$\mathcal{A}$ = {(p,پ,43),(a,ا,321),(a,ε,100),(a,ا‌ی,10),(a,ی,35),(r,ر,201),(i,ی,62),(i,ε,5),(c,س,80),(c,چ,25),(c,ش,1),(t,ت,52)}	

Figure 1: A backparsing example. Note middle tuples in forward and backward parsings are not merged in $\mathcal{A}$ till the alignment is successfully completed.

Method	Intermediate Sequence	Segment(Pattern)	Backoff
Bigram	N/A	#s, sh, he, el, ll, le, ey	s,h,e,l,e,y
CV-MODEL1	$CCVCCV$	sh(CC), hel(CVC), ll(CC), lley($\mathcal{CV}$)	s(C), h(C), e($\mathcal{V}$), l(C), e($\mathcal{V}$), y($\mathcal{V}$)
CV-MODEL2	$CCVCCV$	sh(CC), e(CVC), ll(CC), ey($\overline{\mathcal{CV}}$)	As Above.
CV-MODEL3	$\mathcal{CVCV}$	#sh($\mathcal{C}$), e($\mathcal{CVC}$), ll($\mathcal{C}$), ey($\overline{\mathcal{CV}}$)	sh($\mathcal{C}$), s($\mathcal{C}$), h($\mathcal{C}$), e($\mathcal{V}$), l($\mathcal{C}$), e($\mathcal{V}$), y($\mathcal{V}$)

Figure 2: An example of transliteration for the word pair (shelley,ش‌ل‌ی). Underlined characters are actually transliterated for each segment.

4.1 Baseline

Most transliteration methods reported in the literature — either grapheme- or phoneme-based — use n-grams (AbdulJaleel and Larkey, 2003; Jung et al., 2000). The n-gram-based methods differ mainly in the way that words are segmented, both for training and transliteration generation. A simple n-gram based method works only on single characters (unigram) and transformation rules are defined as $s_i \rightarrow t_j$, while an advanced method may take the surrounding context into account (Jung et al., 2000). We found that using one past symbol (bigram model) works better than other n-gram based methods for English to Persian transliteration (Karimi et al., 2006).

Our collapsed-vowel methods consider language knowledge to improve the string segmentation of n-gram techniques (Karimi et al., 2006). The process begins by generating the *consonant-vowel* sequence (Definition 2) of a source word. For example, the word "shelley" is represented by the sequence $p = CCVCCVV$. Then, following the collapsed vowel concept (Definition 3), this sequence becomes "$CCVCCV$". These approaches, which we refer to as CV-MODEL1 and CV-MODEL2 respectively, partition these sequences using basic patterns (C and $\mathcal{V}$) and main patterns (CC, CVC, $\mathcal{V}C$ and $C\mathcal{V}$). In the training phase, transliteration rules are formed according to the boundaries of the defined patterns and their aligned counterparts (based on a_{e2p} or a_{p2e}) in the target language word T. Similar segmentation is applied during the transliteration generation stage.

4.2 The Proposed Transliteration Approach

The restriction on the context length of consonants imposed by CV-MODEL1 and CV-MODEL2 makes the transliteration of consecutive consonants mapping to a particular character in the target language difficult. For example, "ght" in English maps to only one character in Persian: "ت". Dealing with languages which have different alphabets, and for which the number of characters in their alphabets also differs (such as 26 and 32 for English and Persian), increases the possibility of facing these cases, especially when moving from the language with smaller alphabet size to the one with a larger size. To more effectively address this, we propose a *collapsed consonant and vowel* method (CV-MODEL3) which uses the full reduced sequence (Definition 3), rather than simply reduced vowel sequences. Although recognition of consonant segments is based on the vowel positions, consonants are considered as independent blocks in each string. Conversely, vowels are transliterated in the context of surrounding

consonants, as demonstrated in the example below.

A special symbol is used to indicate the start and/or end of each word if the beginning and end of the word is a consonant respectively. Therefore, for the words starting or ending with consonants, the symbol "#" is added, which is treated as a consonant and therefore grouped in the consonant segment. An example of applying this technique is shown in Figure 2 for the string "shelley". In this example, "sh" and "ll" are treated as two consonant segments, where the transliteration of individual characters inside a segment is dependent on the other members but not the surrounding segments. However, this is not the case for vowel sequences which incorporate a level of knowledge about any segment neighbours. Therefore, for the example "shelley", the first segment is "sh" which belongs to $\mathcal{C}$ pattern. During transliteration, if "#sh" does not appear in any existing rules, a backoff splits the segment to smaller segments: "#" and "sh", or "s"and "h". The second segment contains the vowel "e". Since this vowel is surrounded by consonants, the segment pattern is $\mathcal{CVC}$. In this case, backoff only applies for vowels as consonants are supposed to be part of their own independent segments. That is, if search in the rules of pattern $\mathcal{CVC}$ was unsuccessful, it looks for "e" in $\mathcal{V}$ pattern. Similarly, segmentation for this word continues with "ll" in $\mathcal{C}$ pattern and "ey" in $\mathcal{CV}$ pattern ("y" is an *approximant*, and therefore considered as a vowel when transliterating English to Persian).

4.3 Rules for Back-Transliteration

Written Persian ignores short vowels, and only long vowels appear in text. This causes most English vowels to disappear when transliterating from English to Persian; hence, these vowels must be restored during back-transliteration.

When the initial transliteration happens from English to Persian, the transliterator (whether human or machine) uses the rules of transliterating from English as the source language. Therefore, transliterating back to the original language should consider the original process, to avoid losing essential information. In terms of segmentation in *collapsed-vowel* models, different patterns define segment boundaries in which vowels are necessary clues. Although we do not have most of these vowels in the transliteration generation phase, it is possible to benefit from their existence in the training phase. For example, using CV-MODEL3, the pair (مرکل,merkel) with q_S=$\mathcal{C}$ and a_{p2e}=((م,me),(ر,r),(ک,ke),(ل,l)), produces just one transformation rule "مرکل → merkel" based on a $\mathcal{C}$ pattern. That is, the Persian string contains no vowel characters. If, during the transliteration generation phase, a source word "ممرکل" (S=مرکل) is entered, there would be one and only one output of "merkel", while an alternative such as "mercle" might be required instead. To avoid overfitting the system by long consonant clusters, we perform segmentation based on the English q sequence, but categorise the rules based on their Persian segment counterparts. That is, for the pair (مرکل,merkel) with a_{e2p}=((m,م),(e,ε),(r,ر),(k,ک),(e,ε),(l,ل)), these rules are generated (with category patterns given in parenthesis): م → m ($\mathcal{C}$), کر → rk ($\mathcal{C}$), ل → l ($\mathcal{C}$), کرم → merk ($\mathcal{C}$), لکر → rkel ($\mathcal{C}$). We call the suggested training approach *reverse segmentation*.

Reverse segmentation avoids clustering all the consonants in one rule, since many English words might be transliterated to all-consonant Persian words.

4.4 Transliteration Generation and Ranking

In the transliteration generation stage, the source word is segmented following the same process of segmenting words in training stage, and a probability is computed for each generated target word:

$$P(T|S) = \prod_{k=1}^{|K|} P(\hat{T}_k|\hat{S}_k),$$

where $|K|$ is the number of distinct source segments. $P(\hat{T}_k|\hat{S}_k)$ is the probability of the $\hat{S}_k \rightarrow \hat{T}_k$ transformation rule, as obtained from the training stage:

$$P(\hat{T}_k|\hat{S}_k) = \frac{\text{frequency of } \hat{S}_k \rightarrow \hat{T}_k}{\text{frequency of } \hat{S}_k},$$

where frequency of $\hat{S}_k$ is the number of its occurrence in the transformation rules. We apply a tree structure, following Dijkstra's α-shortest path, to generate the α highest scoring (most probable) transliterations, ranked based on their probabilities.

Corpus		Baseline			CV-MODEL3	
		Bigram	CV-MODEL1	CV-MODEL2	GIZA++	New Alignment
Small Corpus						
	TOP-1	58.0 (2.2)	61.7 (3.0)	60.0 (3.9)	67.4 (5.5)	72.2 (2.2)
	TOP-5	85.6 (3.4)	80.9 (2.2)	86.0 (2.8)	90.9 (2.1)	92.9 (1.6)
	TOP-10	89.4 (2.9)	82.0 (2.1)	91.2 (2.5)	93.8 (2.1)	93.5 (1.7)
Large Corpus						
	TOP-1	47.2 (1.0)	50.6 (2.5)	47.4 (1.0)	55.3 (0.8)	59.8 (1.1)
	TOP-5	77.6 (1.4)	79.8 (3.4)	79.2 (1.0)	84.5 (0.7)	85.4 (0.8)
	TOP-10	83.3 (1.5)	84.9 (3.1)	87.0 (0.9)	89.5 (0.4)	92.6 (0.7)

Table 1: Mean (standard deviation) word accuracy (%) for English to Persian transliteration.

5 Experiments

To investigate the effectiveness of CV-MODEL3 and the new alignment approach on transliteration, we first compare CV-MODEL3 with baseline systems, employing GIZA++ for alignment generation during system training. We then evaluate the same systems, using our new alignment approach. Back-transliteration is also investigated, applying both alignment systems and *reverse segmentation*. In all our experiments, we used ten-fold cross-validation. The statistical significance of different performance levels are evaluated using a paired t-test. The notation TOP-X indicates the first X transliterations prodcued by the automatic methods.

We used two corpora of word pairs in English and Persian: the first, called *Large*, contains 16,670 word pairs; the second, *Small*, contains 1,857 word pairs, and are described fully in our previous paper (Karimi et al., 2006).

The results of transliteration experiments are evaluated using word accuracy (Kang and Choi, 2000) which measures the proportion of transliterations that are correct out of the test corpus.

5.1 Accuracy of Transliteration Approaches

The results of our experiments for transliterating English to Persian, using GIZA++ for alignment generation, are shown in Table 1. CV-MODEL3 outperforms all three baseline systems significantly in TOP-1 and TOP-5 results, for both Persian corpora. TOP-1 results were improved by 9.2% to 16.2% ($p<0.0001$, paired t-test) relative to the baseline systems for the *Small* corpus. For the *Large* corpus, CV-MODEL3 was 9.3% to 17.2% ($p<0.0001$) more accurate relative to the baseline systems.

The results of applying our new alignment algorithm are presented in the last column of Table 1, comparing word accuracy of CV-MODEL3 using GIZA++ and the new alignment for English to Persian transliteration. Transliteration accuracy increases in TOP-1 for both corpora (a relative increase of 7.1% ($p=0.002$) for the *Small* corpus and 8.1% ($p<0.0001$) for the *Large* corpus). The TOP-10 results of the Large corpus again show a relative increase of 3.5% ($p=0.004$). Although the new alignment also increases the performance for TOP-5 and TOP-10 of the *Small* corpus, these increases are not statistically significant.

5.2 Accuracy of Back-Transliteration

The results of back-transliteration are shown in Table 2. We first consider performance improvements gained from using CV-MODEL3: CV-MODEL3 using GIZA++ outperforms Bigram, CV-MODEL1 and CV-MODEL2 by 12.8% to 40.7% ($p<0.0001$) in TOP-1 for the *Small* corpus. The corresponding improvement for the *Large* corpus is 12.8% to 74.2% ($p<0.0001$).

The fifth column of the table shows the performance increase when using CV-MODEL3 with the new alignment algorithm: for the *Large* corpus, the new alignment approach gives a relative increase in accuracy of 15.5% for TOP-5 ($p<0.0001$) and 10% for TOP-10 ($p=0.005$). The new alignment method does not show a significant difference using CV-MODEL3 for the *Small* corpus.

The final column of Table 2 shows the performance of the CV-MODEL3 with the new reverse segmentation approach. Reverse segmentation leads to a significant improvement over the new alignment approach in TOP-1 results for the *Small* corpus by 40.1% ($p<0.0001$), and 49.4% ($p<0.0001$) for the *Large* corpus.

Corpus		Bigram	CV-MODEL1	CV-MODEL2	CV-MODEL3		
					GIZA++	New Alignment	Reverse
Small Corpus							
	TOP-1	23.1 (2.0)	28.8 (4.6)	24.9 (2.8)	32.5 (3.6)	34.4 (3.8)	48.2 (2.9)
	TOP-5	40.8 (3.1)	51.0 (4.8)	52.9 (3.4)	56.0 (3.5)	54.8 (3.7)	68.1 (4.9)
	TOP-10	50.1 (4.1)	58.2 (5.3)	63.2 (3.1)	64.2 (3.2)	63.8 (3.6)	75.7 (4.2)
Large Corpus							
	TOP-1	10.1 (0.6)	15.6 (1.0)	12.0 (1.0)	17.6 (0.8)	18.0 (1.2)	26.9 (0.7)
	TOP-5	20.6 (1.2)	31.7 (0.9)	28.0 (0.7)	36.2 (0.5)	41.8 (1.2)	41.3 (1.7)
	TOP-10	27.2 (1.0)	40.1 (1.1)	37.4 (0.8)	46.0 (0.8)	50.6 (1.1)	49.3 (1.6)

Table 2: Comparison of mean (standard deviation) word accuracy (%) for Persian to English transliteration.

6 Conclusions

We have presented a new algorithm for English to Persian transliteration, and a novel alignment algorithm applicable for transliteration. Our new transliteration method (CV-MODEL3) outperforms the previous approaches for English to Persian, increasing word accuracy by a relative 9.2% to 17.2% (TOP-1), when using GIZA++ for alignment in training. This method shows further 7.1% to 8.1% increase in word accuracy (TOP-1) with our new alignment algorithm.

Persian to English back-transliteration is also investigated, with CV-MODEL3 significantly outperforming other methods. Enriching this model with a new reverse segmentation algorithm gives rise to further accuracy gains in comparison to directly applying English to Persian methods.

In future work we will investigate whether phonetic information can help refine our CV-MODEL3, and experiment with manually constructed rules as a baseline system.

Acknowledgments

This work was supported in part by the Australian government IPRS program (SK) and an ARC Discovery Project Grant (AT).

References

Nasreen AbdulJaleel and Leah S. Larkey. 2003. Statistical transliteration for English-Arabic cross language information retrieval. In *Conference on Information and Knowledge Management*, pages 139–146.

Slaven Bilac and Hozumi Tanaka. 2005. Direct combination of spelling and pronunciation information for robust back-transliteration. In *Conferences on Computational Linguistics and Intelligent Text Processing*, pages 413–424.

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

Michael A. Covington. 1996. An algorithm to align words for historical comparison. *Computational Linguistics*, 22(4):481–496.

Wei Gao, Kam-Fai Wong, and Wai Lam. 2004. Improving transliteration with precise alignment of phoneme chunks and using contextual features. In *Asia Information Retrieval Symposium*, pages 106–117.

Sung Young Jung, Sung Lim Hong, and Eunok Paek. 2000. An English to Korean transliteration model of extended Markov window. In *Conference on Computational Linguistics*, pages 383–389.

Byung-Ju Kang and Key-Sun Choi. 2000. Automatic transliteration and back-transliteration by decision tree learning. In *Conference on Language Resources and Evaluation*, pages 1135–1411.

Sarvnaz Karimi, Andrew Turpin, and Falk Scholer. 2006. English to Persian transliteration. In *String Processing and Information Retrieval*, pages 255–266.

Alexandre Klementiev and Dan Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Association for Computational Linguistics*, pages 817–824.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599–612.

Haizhou Li, Min Zhang, and Jian Su. 2004. A joint source-channel model for machine transliteration. In *Association for Computational Linguistics*, pages 159–166.

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

Jong-Hoon Oh and Key-Sun Choi. 2002. An English-Korean transliteration model using pronunciation and contextual rules. In *Conference on Computational Linguistics*.

Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of proper names in cross-language applications. In *ACM SIGIR Conference on Research and Development on Information Retrieval*, pages 365–366.

Dmitry Zelenko and Chinatsu Aone. 2006. Discriminative methods for transliteration. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing.*, pages 612–617.

Alignment-Based Discriminative String Similarity

Shane Bergsma and Grzegorz Kondrak
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada, T6G 2E8
{bergsma,kondrak}@cs.ualberta.ca

Abstract

A character-based measure of similarity is an important component of many natural language processing systems, including approaches to transliteration, coreference, word alignment, spelling correction, and the identification of cognates in related vocabularies. We propose an alignment-based *discriminative* framework for string similarity. We gather features from substring pairs consistent with a character-based alignment of the two strings. This approach achieves exceptional performance; on nine separate cognate identification experiments using six language pairs, we more than double the precision of traditional orthographic measures like Longest Common Subsequence Ratio and Dice's Coefficient. We also show strong improvements over other recent discriminative and heuristic similarity functions.

1 Introduction

String similarity is often used as a means of quantifying the likelihood that two pairs of strings have the same underlying meaning, based purely on the character composition of the two words. Strube et al. (2002) use Edit Distance as a feature for determining if two words are coreferent. Taskar et al. (2005) use French-English common letter sequences as a feature for discriminative word alignment in bilingual texts. Brill and Moore (2000) learn misspelled-word to correctly-spelled-word similarities for spelling correction. In each of these examples, a similarity measure can make use of the recurrent substring pairings that reliably occur between words having the same meaning.

Across natural languages, these recurrent substring correspondences are found in word pairs known as cognates: words with a common form and meaning across languages. Cognates arise either from words in a common ancestor language (e.g. *light/Licht, night/Nacht* in English/German) or from foreign word borrowings (e.g. *trampoline/toranporin* in English/Japanese). Knowledge of cognates is useful for a number of applications, including sentence alignment (Melamed, 1999) and learning translation lexicons (Mann and Yarowsky, 2001; Koehn and Knight, 2002).

We propose an alignment-based, discriminative approach to string similarity and evaluate this approach on cognate identification. Section 2 describes previous approaches and their limitations. In Section 3, we explain our technique for automatically creating a cognate-identification training set. A novel aspect of this set is the inclusion of *competitive counter-examples* for learning. Section 4 shows how discriminative features are created from a character-based, minimum-edit-distance alignment of a pair of strings. In Section 5, we describe our bitext and dictionary-based experiments on six language pairs, including three based on non-Roman alphabets. In Section 6, we show significant improvements over traditional approaches, as well as significant gains over more recent techniques by Ristad and Yianilos (1998), Tiedemann (1999), Kondrak (2005), and Klementiev and Roth (2006).

2 Related Work

String similarity is a fundamental concept in a variety of fields and hence a range of techniques

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 656–663,
Prague, Czech Republic, June 2007.

have been developed. We focus on approaches that have been applied to words, i.e., uninterrupted sequences of characters found in natural language text. The most well-known measure of the similarity of two strings is the Edit Distance or Levenshtein Distance (Levenshtein, 1966): the number of insertions, deletions and substitutions required to transform one string into another. In our experiments, we use Normalized Edit Distance (NED): Edit Distance divided by the length of the longer word. Other popular measures include Dice's Coefficient (DICE) (Adamson and Boreham, 1974), and the length-normalized measures Longest Common Subsequence Ratio (LCSR) (Melamed, 1999), and Longest Common Prefix Ratio (PREFIX) (Kondrak, 2005). These baseline approaches have the important advantage of not requiring training data. We can also include in the non-learning category Kondrak (2005)'s Longest Common Subsequence Formula (LCSF), a probabilistic measure designed to mitigate LCSR's preference for shorter words.

Although simple to use, the untrained measures cannot adapt to the specific spelling differences between a pair of languages. Researchers have therefore investigated adaptive measures that are learned from a set of known cognate pairs. Ristad and Yianilos (1998) developed a stochastic transducer version of Edit Distance learned from unaligned string pairs. Mann and Yarowsky (2001) saw little improvement over Edit Distance when applying this transducer to cognates, even when filtering the transducer's probabilities into different weight classes to better approximate Edit Distance. Tiedemann (1999) used various measures to learn the recurrent spelling changes between English and Swedish, and used these changes to re-weight LCSR to identify more cognates, with modest performance improvements. Mulloni and Pekar (2006) developed a similar technique to improve NED for English/German.

Essentially, all these techniques improve on the baseline approaches by using a set of positive (true) cognate pairs to re-weight the costs of edit operations or the score of sequence matches. Ideally, we would prefer a more flexible approach that can learn positive *or* negative weights on *substring* pairings in order to better identify related strings. One system that can potentially provide this flexibility is a discriminative string-similarity approach to named-entity transliteration by Klementiev and Roth (2006). Although not compared to other similarity measures in the original paper, we show that this discriminative technique can strongly outperform traditional methods on cognate identification.

Unlike many recent generative systems, the Klementiev and Roth approach does not exploit the known positions in the strings where the characters match. For example, Brill and Moore (2000) combine a character-based alignment with the Expectation Maximization (EM) algorithm to develop an improved probabilistic error model for spelling correction. Rappoport and Levent-Levi (2006) apply this approach to learn substring correspondences for cognates. Zelenko and Aone (2006) recently showed a Klementiev and Roth (2006)-style discriminative approach to be superior to alignment-based generative techniques for name transliteration. Our work successfully uses the alignment-based methodology of the generative approaches to enhance the feature set for discriminative string similarity.

3 The Cognate Identification Task

Given two string lists, E and F, the task of cognate identification is to find all pairs of strings (e, f) that are cognate. In other similarity-driven applications, E and F could be misspelled and correctly spelled words, or the orthographic and the phonetic representation of words, etc. The task remains to link strings with common meaning in E and F using only the string similarity measure.

We can facilitate the application of string similarity to cognates by using a definition of cognation not dependent on etymological analysis. For example, Mann and Yarowsky (2001) define a word pair (e, f) to be cognate if they are a translation pair (same meaning) and their Edit Distance is less than three (same form). We adopt an improved definition (suggested by Melamed (1999) for the French-English Canadian Hansards) that does not over-propose shorter word pairs: (e, f) are cognate if they are translations and their LCSR $\geq$ 0.58. Note that this cutoff is somewhat conservative: the English/German cognates *light/Licht* (LCSR=0.8) are included, but not the cognates *eight/acht* (LCSR=0.4).

If two words must have LCSR $\geq$ 0.58 to be cog-

Foreign Language F	Words $f \in F$	Cognates E_{f+}	False Friends E_{f-}
Japanese (Rômaji)	napukin	napkin	nanking, pumpkin, snacking, sneaking
French	abondamment	abundantly	abandonment, abatement, ... wonderment
German	prozyklische	procyclical	polished, prophylactic, prophylaxis

Table 1: Foreign-English cognates and false friend training examples.

nate, then for a given word $f \in F$, we need only consider as possible cognates the subset of words in E having an LCSR with f larger than 0.58, a set we call E_f. The portion of E_f with the same meaning as f, E_{f+}, are cognates, while the part with different meanings, E_{f-}, are not cognates. The words E_{f-} with similar spelling but different meaning are sometimes called *false friends*. The cognate identification task is, for every word $f \in F$, and a list of similarly spelled words E_f, to distinguish the cognate subset E_{f+} from the false friend set E_{f-}.

To create training data for our learning approaches, and to generate a high-quality labelled test set, we need to annotate some of the $(f, e_f \in E_f)$ word pairs for whether or not the words share a common meaning. In Section 5, we explain our two high-precision automatic annotation methods: checking if each pair of words (a) were aligned in a word-aligned bitext, or (b) were listed as translation pairs in a bilingual dictionary.

Table 1 provides some labelled examples with non-empty cognate and false friend lists. Note that despite these examples, this is not a ranking task: even in highly related languages, most words in F have empty E_{f+} lists, and many have empty E_{f-} as well. Thus one natural formulation for cognate identification is a pairwise (and symmetric) cognation classification that looks at each pair (f, e_f) separately and individually makes a decision:

+(napukin,napkin)
– (napukin,nanking)
– (napukin,pumpkin)

In this formulation, the benefits of a discriminative approach are clear: it must find substrings that distinguish cognate pairs from word pairs with otherwise similar form. Klementiev and Roth (2006), although using a discriminative approach, do not provide their infinite-attribute perceptron with competitive counter-examples. They instead use transliterations as positives and randomly-paired English and Russian words as negative examples. In the following section, we also improve on Klementiev and Roth (2006) by using a character-based string alignment to focus the features for discrimination.

4 Features for Discriminative Similarity

Discriminative learning works by providing a training set of labelled examples, each represented as a set of features, to a module that learns a classifier. In the previous section we showed how labelled word pairs can be collected. We now address methods of representing these word pairs as sets of features useful for determining cognation.

Consider the Rômaji Japanese/English cognates: *(sutoresu,stress)*. The LCSR is 0.625. Note that the LCSR of *sutoresu* with the English false friend *stories* is higher: 0.75. LCSR alone is too weak a feature to pick out cognates. We need to look at the actual character substrings.

Klementiev and Roth (2006) generate features for a pair of words by splitting both words into all possible substrings of up to size two:
sutoresu $\Rightarrow$ { *s, u, t, o, r, e, s, u, su, ut, to, ... su* }
stress $\Rightarrow$ { *s, t, r, e, s, s, st, tr, re, es, ss* }
Then, a feature vector is built from all substring pairs from the two words such that the difference in positions of the substrings is within one:
{*s-s, s-t, s-st, su-s, su-t, su-st, su-tr... r-s, r-s, r-es...*}
This feature vector provides the feature representation used in supervised machine learning.

This example also highlights the limitations of the Klementiev and Roth approach. The learner can provide weight to features like *s-s* or *s-st* at the beginning of the word, but because of the gradual accumulation of positional differences, the learner never sees the *tor-tr* and *es-es* correspondences that really help indicate the words are cognate.

Our solution is to use the minimum-edit-distance alignment of the two strings as the basis for feature extraction, rather than the positional correspondences. We also include beginning-of-word (^) and end-of-word ($) markers (referred to as *boundary*

markers) to highlight correspondences at those positions. The pair $(sutoresu, stress)$ can be aligned:

For the feature representation, we only extract substring pairs that are consistent with this alignment.[1] That is, the letters in our pairs can only be aligned to each other and not to letters outside the pairing:
{ *ˆ-ˆ, ˆs-ˆs, s-s, su-s, ut-t, t-t,... es-es, s-s, su-ss...*}
We define *phrase* pairs to be the pairs of substrings consistent with the alignment. A similar use of the term "phrase" exists in machine translation, where phrases are often pairs of word sequences consistent with word-based alignments (Koehn et al., 2003).

By limiting the substrings to only those pairs that are consistent with the alignment, we generate fewer, more-informative features. Using more precise features allows a larger maximum substring size L than is feasible with the positional approach. Larger substrings allow us to capture important recurring deletions like the "u" in *sut-st*.

Tiedemann (1999) and others have shown the importance of using the mismatching portions of cognate pairs to learn the recurrent spelling changes between two languages. In order to capture mismatching segments longer than our maximum substring size will allow, we include special features in our representation called *mismatches*. *Mismatches* are phrases that span the entire sequence of unaligned characters between two pairs of aligned end characters (similar to the "rules" extracted by Mulloni and Pekar (2006)). In the above example, *su$-ss$* is a mismatch with "s" and "$" as the aligned end characters. Two sets of features are taken from each mismatch, one that includes the beginning/ending aligned characters as context and one that does not. For example, for the endings of the French/English pair *(économique,economic)*, we include both the substring pairs *ique$:ic$* and *que:c* as features.

One consideration is whether substring features should be binary presence/absence, or the count of the feature in the pair normalized by the length of the longer word. We investigate both of these approaches in our experiments. Also, there is no reason not to include the scores of baseline approaches like NED, LCSR, PREFIX or DICE as features in the representation as well. Features like the lengths of the two words and the difference in lengths of the words have also proved to be useful in preliminary experiments. Semantic features like frequency similarity or contextual similarity might also be included to help determine cognation between words that are not present in a translation lexicon or bitext.

5 Experiments

Section 3 introduced two high-precision methods for generating labelled cognate pairs: using the word alignments from a bilingual corpus or using the entries in a translation lexicon. We investigate both of these methods in our experiments. In each case, we generate sets of labelled word pairs for training, testing, and development. The proportion of positive examples in the bitext-labelled test sets range between 1.4% and 1.8%, while ranging between 1.0% and 1.6% for the dictionary data.[2]

For the discriminative methods, we use a popular Support Vector Machine (SVM) learning package called SVM^{light} (Joachims, 1999). SVMs are maximum-margin classifiers that achieve good performance on a range of tasks. In each case, we learn a linear kernel on the training set pairs and tune the parameter that trades-off training error and margin on the development set. We apply our classifier to the test set and score the pairs by their positive distance from the SVM classification hyperplane (also done by Bilenko and Mooney (2003) with their token-based SVM similarity measure).

We also score the test sets using traditional orthographic similarity measures PREFIX, DICE, LCSR, and NED, an average of these four, and Kondrak (2005)'s LCSF. We also use the log of the edit probability from the stochastic decoder of Ristad and Yianilos (1998) (normalized by the length of the longer word) and Tiedemann (1999)'s highest performing system (Approach #3). Both use only the positive examples in our training set. Our evaluation metric is 11-pt average precision on the score-sorted pair lists (also used by Kondrak and Sherif (2006)).

[1] If the words are from different alphabets, we can get the alignment by mapping the letters to their closest Roman equivalent, or by using the EM algorithm to learn the edits (Ristad and Yianilos, 1998).

[2] The cognate data sets used in our experiments are available at http://www.cs.ualberta.ca/˜bergsma/Cognates/

5.1 Bitext Experiments

For the bitext-based annotation, we use publicly-available word alignments from the Europarl corpus, automatically generated by GIZA++ for French-English (Fr), Spanish-English (Es) and German-English (De) (Koehn and Monz, 2006). Initial cleaning of these noisy word pairs is necessary. We thus remove all pairs with numbers, punctuation, a capitalized English word, and all words that occur fewer than ten times. We also remove many incorrectly aligned words by filtering pairs where the pairwise Mutual Information between the words is less than 7.5. This processing leaves vocabulary sizes of 39K for French, 31K for Spanish, and 60K for German.

Our labelled set is then generated from pairs with LCSR ≥ 0.58 (using the cutoff from Melamed (1999)). Each labelled set entry is a triple of a) the foreign word f, b) the cognates E_{f+} and c) the false friends E_{f-}. For each language pair, we randomly take 20K triples for training, 5K for development and 5K for testing. Each triple is converted to a set of pairwise examples for learning and classification.

5.2 Dictionary Experiments

For the dictionary-based cognate identification, we use French, Spanish, German, Greek (Gr), Japanese (Jp), and Russian (Rs) to English translation pairs from the Freelang program.[3] The latter three pairs were chosen so that we can evaluate on more distant languages that use non-Roman alphabets (although the Rômaji Japanese is Romanized by definition). We take 10K labelled-set triples for training, 2K for testing and 2K for development.

The baseline approaches and our definition of cognation require comparison in a common alphabet. Thus we use a simple context-free mapping to convert every Russian and Greek character in the word pairs to their nearest Roman equivalent. We then label a translation pair as cognate if the LCSR between the words' Romanized representations is greater than 0.58. We also operate all of our comparison systems on these Romanized pairs.

6 Results

We were interested in whether our working definition of cognation (translations and LCSR ≥ 0.58) reflects true etymological relatedness. We looked at the LCSR histogram for translation pairs in one of our translation dictionaries (Figure 1). The trendline suggests a bimodal distribution, with two distinct distributions of translation pairs making up the dictionary: incidental letter agreement gives low LCSR for the larger, non-cognate portion and high LCSR characterizes the likely cognates. A threshold of 0.58 captures most of the cognate distribution while excluding non-cognate pairs. This hypothesis was confirmed by checking the LCSR values of a list of known French-English cognates (randomly collected from a dictionary for another project): 87.4% were above 0.58. We also checked cognation on 100 randomly-sampled, positively-labelled French-English pairs (i.e. translated or aligned and having LCSR ≥ 0.58) from both the dictionary and bitext data. 100% of the dictionary pairs and 93% of the bitext pairs were cognate.

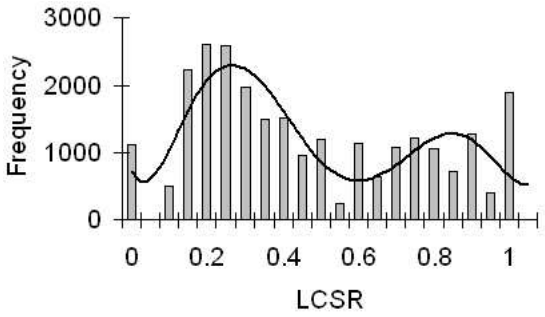


Figure 1: LCSR histogram and polynomial trendline of French-English dictionary pairs.

System	Prec
Klementiev-Roth (KR) L≤2	58.6
KR L≤2 (normalized, boundary markers)	62.9
phrases L≤2	61.0
phrases L≤3	65.1
phrases L≤3 + *mismatches*	65.6
phrases L≤3 + *mismatches* + NED	**65.8**

Table 2: Bitext French-English *development set* cognate identification 11-pt average precision (%).

Next, we investigate various configurations of the discriminative systems on one of our cognate identification development sets (Table 2). The original Klementiev and Roth (2006) (KR) system can

[3] http://www.freelang.net/dictionary/

	Bitext			Dictionary					
System	Fr	Es	De	Fr	Es	De	Gr	Jp	Rs
PREFIX	34.7	27.3	36.3	45.5	34.7	25.5	28.5	16.1	29.8
DICE	33.7	28.2	33.5	44.3	33.7	21.3	30.6	20.1	33.6
LCSR	34.0	28.7	28.5	48.3	36.5	18.4	30.2	24.2	36.6
NED	36.5	**31.9**	32.3	50.1	**40.3**	23.3	**33.9**	28.2	41.4
PREFIX+DICE+LCSR+NED	**38.7**	31.8	**39.3**	**51.6**	40.1	**28.6**	33.7	22.9	37.9
Kondrak (2005): LCSF	29.8	28.9	29.1	39.9	36.6	25.0	30.5	**33.4**	**45.5**
Ristad & Yanilos (1998)	37.7	32.5	34.6	56.1	46.9	36.9	38.0	52.7	51.8
Tiedemann (1999)	38.8	33.0	34.7	55.3	49.0	24.9	37.6	33.9	45.8
Klementiev & Roth (2006)	61.1	55.5	53.2	73.4	62.3	48.3	51.4	62.0	64.4
Alignment-Based Discriminative	**66.5**	**63.2**	**64.1**	**77.7**	**72.1**	**65.6**	**65.7**	**82.0**	**76.9**

Table 3: Bitext, Dictionary Foreign-to-English cognate identification 11-pt average precision (%).

be improved by normalizing the feature count by the longer string length and including the boundary markers. This is therefore done with all the alignment-based approaches. Also, because of the way its features are constructed, the KR system is limited to a maximum substring length of two (L≤2). A maximum length of three (L≤3) in the KR framework produces millions of features and prohibitive training times, while L≤3 is computationally feasible in the phrasal case, and increases precision by 4.1% over the *phrases L≤2* system.[4] Including *mismatches* results in another small boost in performance (0.5%), while using an Edit Distance feature again increases performance by a slight margin (0.2%). This ranking of configurations is consistent across all the bitext-based development sets; we therefore take the configuration of the highest scoring system as our Alignment-Based Discriminative system for the remainder of this paper.

We next compare the Alignment-Based Discriminative scorer to the various other implemented approaches across the three bitext and six dictionary-based cognate identification test sets (Table 3). The table highlights the top system among both the non-adaptive and adaptive similarity scorers.[5] In each language pair, the alignment-based discriminative approach outperforms all other approaches, but the KR system also shows strong gains over non-adaptive techniques and their re-weighted extensions. This is in contrast to previous comparisons which have only demonstrated minor improvements with adaptive over traditional similarity measures (Kondrak and Sherif, 2006).

We consistently found that the original KR performance could be surpassed by a system that normalizes the KR feature count and adds boundary markers. Across all the test sets, this modification results in a 6% average gain in performance over baseline KR, but is still on average 5% below the Alignment-Based Discriminative technique, with a statistically significantly difference on each of the nine sets.[6]

Figure 2 shows the relationship between training data size and performance in our bitext-based French-English data. Note again that the Tiedemann and Ristad & Yanilos systems only use the positive examples in the training data. Our alignment-based similarity function outperforms all the other systems across nearly the entire range of training data. Note also that the discriminative learning curves show no signs of slowing down: performance grows logarithmically from 1K to 846K word pairs.

For insight into the power of our discriminative approach, we provide some of our classifiers' highest and lowest-weighted features (Table 4).

[4]Preliminary experiments using even longer phrases (beyond L≤3) currently produce a computationally prohibitive number of features for SVM learning. Deploying current feature selection techniques might enable the use of even more expressive and powerful feature sets with longer phrase lengths.

[5]Using the training data and the SVM to weight the components of the PREFIX+DICE+LCSR+NED scorer resulted in negligible improvements over the simple average on our development data.

[6]Following Evert (2004), significance was computed using Fisher's exact test (at $p = 0.05$) to compare the n-best word pairs from the scored test sets, where n was taken as the number of positive pairs in the set.

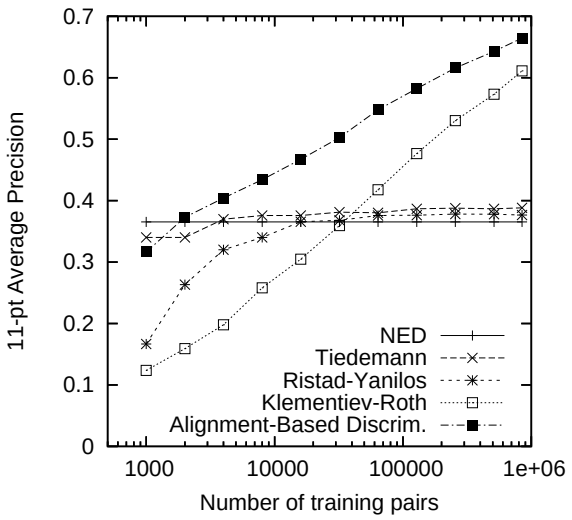


Figure 2: Bitext French-English cognate identification learning curve.

Lang.	Feat.	Wt.	Example
Fr (Bitext)	ées-ed	+8.0	vérifiées:verified
Jp (Dict.)	ru-l	+5.9	penaruti:penalty
De (Bitext)	k-c	+5.5	kreativ:creative
Rs (Dict.)	irov-_	+4.9	motivirovat:motivate
Gr (Dict.)	f-ph	+4.1	symfonia:symphony
Gr (Dict.)	kos-c	+3.3	anarchikos:anarchic
Gr (Dict.)	os$-y$	-2.5	*anarchikos:anarchy*
Jp (Dict.)	ou-ou	-2.6	*handoutai:handout*
Es (Dict.)	_-un	-3.1	*balance:unbalance*
Fr (Dict.)	er$-er$	-5.0	*former:former*
Es (Bitext)	mos-s	-5.1	*toleramos:tolerates*

Table 4: Example features and weights for various Alignment-Based Discriminative classifiers (Foreign-English, negative pairs in *italics*).

Note the expected correspondences between foreign spellings and English (*k-c*, *f-ph*), but also features that leverage derivational and inflectional morphology. For example, Greek-English pairs with the adjective-ending correspondence *kos-c*, e.g. *anarchikos:anarchic*, are favoured, but pairs with the adjective ending in Greek and noun ending in English, *os$-y$*, are penalized; indeed, by our definition, *anarchikos:anarchy* is not cognate. In a bitext, the feature *ées-ed* captures that feminine-plural inflection of past tense verbs in French corresponds to regular past tense in English. On the other hand, words ending in the Spanish first person plural verb suffix *-amos* are rarely translated to English words ending with the suffix *-s*, causing *mos-s* to be penalized. The ability to leverage negative features, learned from appropriate counter examples, is a key innovation of our discriminative framework.

Gr-En (*Dict.*)	Es-En (*Bitext*)
alkali:alkali	agenda:agenda
makaroni:macaroni	natural:natural
adrenalini:adrenaline	márgenes:margins
flamingko:flamingo	hormonal:hormonal
spasmodikos:spasmodic	radón:radon
amvrosia:ambrosia	higiénico:hygienic

Table 5: Highest scored pairs by Alignment-Based Discriminative classifier (negative pairs in *italics*).

Table 5 gives the top pairs scored by our system on two of the sets. Notice that unlike traditional similarity measures that always score identical words higher than all other pairs, by virtue of our feature weighting, our discriminative classifier prefers some pairs with very characteristic spelling changes.

We performed error analysis by looking at all the pairs our system scored quite confidently (highly positive or highly negative similarity), but which were labelled oppositely. Highly-scored false positives arose equally from 1) actual cognates not linked as translations in the data, 2) related words with diverged meanings, e.g. the error in Table 5: *makaroni* in Greek actually means *spaghetti* in English, and 3) the same word stem, a different part of speech (e.g. the Greek/English adjective/noun *synonymos:synonym*). Meanwhile, inspection of the highly-confident false negatives revealed some (often erroneously-aligned in the bitext) positive pairs with incidental letter match (e.g. the French/English *recettes:proceeds*) that we would not actually deem to be cognate. Thus the errors that our system makes are often either linguistically interesting or point out mistakes in our automatically-labelled bitext and (to a lesser extent) dictionary data.

7 Conclusion

This is the first research to apply discriminative string similarity to the task of cognate identification. We have introduced and successfully applied an alignment-based framework for discriminative similarity that consistently demonstrates improved performance in both bitext and dictionary-based cog-

nate identification on six language pairs. Our improved approach can be applied in any of the diverse applications where traditional similarity measures like Edit Distance and LCSR are prevalent. We have also made available our cognate identification data sets, which will be of interest to general string similarity researchers.

Furthermore, we have provided a natural framework for future cognate identification research. Phonetic, semantic, or syntactic features could be included within our discriminative infrastructure to aid in the identification of cognates in text. In particular, we plan to investigate approaches that do not require the bilingual dictionaries or bitexts to generate training data. For example, researchers have automatically developed translation lexicons by seeing if words from each language have similar frequencies, contexts (Koehn and Knight, 2002), burstiness, inverse document frequencies, and date distributions (Schafer and Yarowsky, 2002). Semantic and string similarity might be learned jointly with a co-training or bootstrapping approach (Klementiev and Roth, 2006). We may also compare alignment-based discriminative string similarity with a more complex discriminative model that learns the alignments as latent structure (McCallum et al., 2005).

Acknowledgments

We gratefully acknowledge support from the Natural Sciences and Engineering Research Council of Canada, the Alberta Ingenuity Fund, and the Alberta Informatics Circle of Research Excellence.

References

George W. Adamson and Jillian Boreham. 1974. The use of an association measure based on character structure to identify semantically related pairs of words and document titles. *Information Storage and Retrieval*, 10:253–260.

Mikhail Bilenko and Raymond J. Mooney. 2003. Adaptive duplicate detection using learnable string similarity measures. In *KDD*, pages 39–48.

Eric Brill and Robert Moore. 2000. An improved error model for noisy channel spelling correction. In *ACL*. 286–293.

Stefan Evert. 2004. Significance tests for the evaluation of ranking methods. In *COLING*, pages 945–951.

Thorsten Joachims. 1999. Making large-scale Support Vector Machine learning practical. In *Advances in Kernel Methods: Support Vector Machines*, pages 169–184. MIT-Press.

Alexandre Klementiev and Dan Roth. 2006. Named entity transliteration and discovery from multilingual comparable corpora. In *HLT-NAACL*, pages 82–88.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *ACL Workshop on Unsupervised Lexical Acquistion*.

Philipp Koehn and Christof Monz. 2006. Manual and automatic evaluation of machine translation between European languages. In *NAACL Workshop on Statistical Machine Translation*, pages 102–121.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *HLT-NAACL*, pages 127–133.

Grzegorz Kondrak and Tarek Sherif. 2006. Evaluation of several phonetic similarity algorithms on the task of cognate identification. In *COLING-ACL Workshop on Linguistic Distances*, pages 37–44.

Grzegorz Kondrak. 2005. Cognates and word alignment in bitexts. In *MT Summit X*, pages 305–312.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Gideon S. Mann and David Yarowsky. 2001. Multipath translation lexicon induction via bridge languages. In *NAACL*, pages 151–158.

Andrew McCallum, Kedar Bellare, and Fernando Pereira. 2005. A conditional random field for discriminatively-trained finite-state string edit distance. In *UAI*. 388–395.

I. Dan Melamed. 1999. Bitext maps and alignment via pattern recognition. *Computational Linguistics*, 25(1):107–130.

Andrea Mulloni and Viktor Pekar. 2006. Automatic detection of orthographic cues for cognate recognition. In *LREC*, pages 2387–2390.

Ari Rappoport and Tsahi Levent-Levi. 2006. Induction of cross-language affix and letter sequence correspondence. In *EACL Workshop on Cross-Language Knowledge Induction*.

Eric Sven Ristad and Peter N. Yianilos. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532.

Charles Schafer and David Yarowsky. 2002. Inducing translation lexicons via diverse similarity measures and bridge languages. In *CoNLL*, pages 207–216.

Michael Strube, Stefan Rapp, and Christoph Müller. 2002. The influence of minimum edit distance on reference resolution. In *EMNLP*, pages 312–319.

Ben Taskar, Simon Lacoste-Julien, and Dan Klein. 2005. A discriminative matching approach to word alignment. In *HLT-EMNLP*, pages 73–80.

Jörg Tiedemann. 1999. Automatic construction of weighted string similarity measures. In *EMNLP-VLC*, pages 213–219.

Dmitry Zelenko and Chinatsu Aone. 2006. Discriminative methods for transliteration. In *EMNLP*, pages 612–617.

Bilingual Terminology Mining – Using Brain, not brawn comparable corpora

E. Morin, B. Daille
Université de Nantes
LINA FRE CNRS 2729
2, rue de la Houssinière
BP 92208
F-44322 Nantes Cedex 03
{morin-e,daille-b}@
univ-nantes.fr

K. Takeuchi
Okayama University
3-1-1, Tsushimanaka
Okayama-shi, Okayama,
700-8530, Japan
koichi@
cl.it.okayama-u.ac.jp

K. Kageura
Graduate School of Education
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku,
Tokyo, 113-0033, Japan
kyo@p.u-tokyo.ac.jp

Abstract

Current research in text mining favours the quantity of texts over their quality. But for bilingual terminology mining, and for many language pairs, large comparable corpora are not available. More importantly, as terms are defined vis-à-vis a specific domain with a restricted register, it is expected that the quality rather than the quantity of the corpus matters more in terminology mining. Our hypothesis, therefore, is that the quality of the corpus is more important than the quantity and ensures the quality of the acquired terminological resources. We show how important the type of discourse is as a characteristic of the comparable corpus.

1 Introduction

Two main approaches exist for compiling corpora: "Big is beautiful" or "Insecurity in large collections". Text mining research commonly adopts the first approach and favors data quantity over quality. This is normally justified on the one hand by the need for large amounts of data in order to make use of statistic or stochastic methods (Manning and Schütze, 1999), and on the other by the lack of operational methods to automatize the building of a corpus answering to selected criteria, such as domain, register, media, style or discourse.

For lexical alignment from comparable corpora, good results on single words can be obtained from large corpora — several millions words — the accuracy of proposed translation is about 80% for the top 10-20 candidates (Fung, 1998; Rapp, 1999; Chiao and Zweigenbaum, 2002). (Cao and Li, 2002) have achieved 91% accuracy for the top three candidates using the Web as a comparable corpus. But for specific domains, and many pairs of languages, such huge corpora are not available. More importantly, as terms are defined vis-à-vis a specific domain with a restricted register, it is expected that the quality rather than the quantity of the corpus matters more in terminology mining. For terminology mining, therefore, our hypothesis is that the quality of the corpora is more important than the quantity and that this ensures the quality of the acquired terminological resources.

Comparable corpora are "sets of texts in different languages, that are not translations of each other" (Bowker and Pearson, 2002, p. 93). The term *comparable* is used to indicate that these texts share some characteristics or features: topic, period, media, author, register (Biber, 1994), discourse... This corpus comparability is discussed by lexical alignment researchers but never demonstrated: it is often reduced to a specific domain, such as the medical (Chiao and Zweigenbaum, 2002) or financial domains (Fung, 1998), or to a register, such as newspaper articles (Fung, 1998). For terminology

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 664–671,
Prague, Czech Republic, June 2007.

mining, the comparability of the corpus should be based on the domain or the sub-domaine, but also on the type of discourse. Indeed, discourse acts semantically upon the lexical units. For a defined topic, some terms are specific to one discourse or another. For example, for French, within the sub-domain of obesity in the domain of medicine, we find the term *excès de poids* (overweight) only inside texts sharing a popular science discourse, and the synonym *excès pondéral* (overweight) only in scientific discourse. In order to evaluate how important the discourse criterion is for building bilingual terminological lists, we carried out experiments on French-Japanese comparable corpora in the domain of medicine, more precisely on the topic of diabetes and nutrition, using texts collected from the Web and manually selected and classified into two discourse categories: one contains only scientific documents and the other contains both scientific and popular science documents.

We used a state-of-the-art multilingual terminology mining chain composed of two term extraction programs, one in each language, and an alignment program. The term extraction programs are publicly available and both extract multi-word terms that are more precise and specific to a particular scientific domain than single word terms. The alignment program makes use of the direct context-vector approach (Fung, 1998; Peters and Picchi, 1998; Rapp, 1999) slightly modified to handle both single- and multi-word terms. We evaluated the candidate translations of multi-word terms using a reference list compiled from publicly available resources. We found that taking discourse type into account resulted in candidate translations of a better quality even when the corpus size is reduced by half. Thus, even using a state-of-the-art alignment method well-known as data greedy, we reached the conclusion that the quantity of data is not sufficient to obtain a terminological list of high quality and that a real comparability of corpora is required.

2 Multilingual terminology mining chain

Taking as input a comparable corpora, the multilingual terminology chain outputs a list of single- and multi-word candidate terms along with their candidate translations. Its architecture is summarized in Figure 1 and comprises term extraction and alignment programs.

2.1 Term extraction programs

The terminology extraction programs are available for both French[1] (Daille, 2003) and Japanese[2] (Takeuchi et al., 2004). The terminological units that are extracted are multi-word terms whose syntactic patterns correspond either to a canonical or a variation structure. The patterns are expressed using part-of-speech tags: for French, Brill's POS tagger[3] and the FLEM lemmatiser[4] are utilised, and for Japanese, CHASEN[5]. For French, the main patterns are N N, N Prep N et N Adj and for Japanese, N N, N Suff, Adj N and Pref N. The variants handled are morphological for both languages, syntactical only for French, and compounding only for Japanese. We consider as a morphological variant a morphological modification of one of the components of the base form, as a syntactical variant the insertion of another word into the components of the base form, and as a compounding variant the agglutination of another word to one of the components of the base form. For example, in French, the candidate MWT *sécrétion d'insuline* (insulin secretion) appears in the following forms:

- **base form** of N Prep N pattern: *sécrétion d'insuline* (insulin secretion);
- **inflexional variant**: *sécrétions d'insuline* (insulin secretions);
- **syntactic variant** (insertion inside the base form of a modifier): *sécrétion pancréatique d'insuline* (pancreatic insulin secretion);
- **syntactic variant** (expansion coordination of base form): *secrétion de peptide et d'insuline* (insulin and peptide secretion).

The MWT candidates *secrétion insulinique* (insulin secretion) and *hypersécrétion insulinique* (insulin

[1] http://www.sciences.univ-nantes.fr/info/perso/permanents/daille/ and release LINUX.

[2] http://research.nii.ac.jp/~koichi/study/hotal/

[3] http://www.atilf.fr/winbrill/

[4] http://www.univ-nancy2.fr/pers/namer/

[5] http://chasen.org/$\sim$taku/software/mecab/

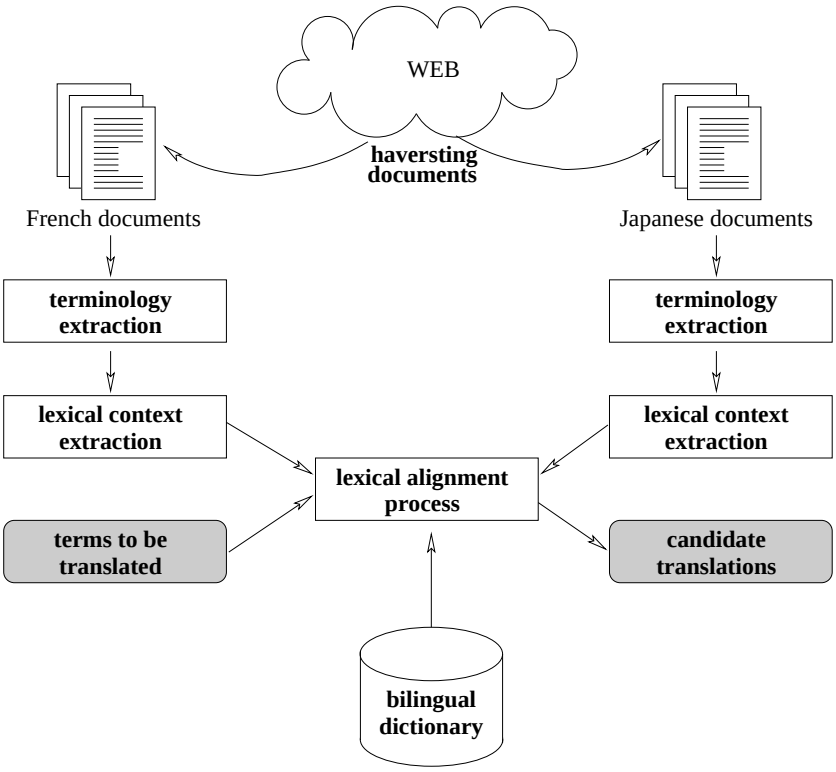


Figure 1: Architecture of the multilingual terminology mining chain

hypersecretion) have also been identified and lead together with *sécrétion d'insuline* (insulin secretion) to a cluster of semantically linked MWTs.

In Japanese, the MWT インスリン.分泌[6] (insulin secretion) appears in the following forms:

- **base form** of NN pattern: インスリン/N_1.分泌/N_2 (insulin secretion);
- **compounding variant** (agglutination of a word at the end of the base form): インスリン/N_1.分泌/N_2.能力/N_3 (insulin secretion ability)

At present, the Japanese term extraction program does not cluster terms.

2.2 Term alignment

The lexical alignment program adapts the direct context-vector approach proposed by (Fung, 1998) for single-word terms (SWTs) to multi-word terms (MWTs). It aligns source MWTs with target single words, SWTs or MWTs. From now on, we will refer to lexical units as words, SWTs or MWTs.

2.2.1 Implementation of the direct context-vector method

Our implementation of the direct context-vector method consists of the following 4 steps:

1. We collect all the lexical units in the context of each lexical unit i and count their occurrence frequency in a window of n words around i. For each lexical unit i of the source and the target language, we obtain a context vector v_i which gathers the set of co-occurrence units j associated with the number of times that j and i occur together occ_j^i. We normalise context vectors using an association score such as Mutual Information or Log-likelihood. In order to reduce the arity of context vectors, we keep only the co-occurrences with the highest association scores.

2. Using a bilingual dictionary, we translate the lexical units of the source context vector.

[6] For all Japanese examples, we explicitly segment the compound into its component parts through the use of the "." symbol.

3. For a word to be translated, we compute the similarity between the translated context vector and all target vectors through vector distance measures such as Cosine (Salton and Lesk, 1968) or Jaccard (Tanimoto, 1958).

4. The candidate translations of a lexical unit are the target lexical units closest to the translated context vector according to vector distance.

2.2.2 Translation of lexical units

The translation of the lexical units of the context vectors, which depends on the coverage of the bilingual dictionary vis-à-vis the corpus, is an important step of the direct approach: more elements of the context vector are translated more the context vector will be discrimating for selecting translations in the target language. If the bilingual dictionary provides several translations for a lexical unit, we consider all of them but weight the different translations by their frequency in the target language. If an MWT cannot be directly translated, we generate possible translations by using a compositional method (Grefenstette, 1999). For each element of the MWT found in the bilingual dictionary, we generate all the translated combinations identified by the term extraction program. For example, in the case of the MWT *fatigue chronique* (chronic fatigue), we have the following four translations for *fatigue*: 疲れ, 疲労, 倦怠, 飽き and the following two translations for *chronique*: 記事番組, 慢性. Next, we generate all combinations of translated elements (See Table 1[7]) and select those which refer to an existing MWT in the target language. Here, only one term has been identified by the Japanese terminology extraction program: 慢性.疲労. In this approach, when it is not possible to translate all parts of an MWT, or when the translated combinations are not identified by the term extraction program, the MWT is not taken into account in the translation process.

This approach differs from that used by (Robitaille et al., 2006) for French/Japanese translation. They first decompose the French MWT into combinations of shorter multi-word units (MWU) elements. This approach makes the direct translation of a subpart of the MWT possible if it is present in the bilingual dictionary. For an MWT of length n, (Robitaille et al., 2006) produce all the combinations of MWU elements of a length less than or equal to n. For example, the French term *syndrome de fatigue chronique* (chronic fatigue disease) yields the following four combinations: i) [*syndrome de fatigue chronique*], ii) [*syndrome de fatigue*] [*chronique*], iii) [*syndrome*] [*fatigue chronique*] and iv) [*syndrome*] [*fatigue*] [*chronique*]. We limit ourselves to the combination of type iv) above since 90% of the candidate terms provided by the term extraction process, after clustering, are only composed of two content words.

chronique	*fatigue*
記事番組	疲れ
慢性	疲れ
記事番組	疲労
慢性	疲労
記事番組	倦怠
慢性	倦怠
記事番組	飽き
慢性	飽き

Table 1: Illustration of the compositional method. The underlined Japanese MWT actually exists.

3 Linguistic resources

In this section we outline the different textual resources used for our experiments: the comparable corpora, bilingual dictionary and reference lexicon.

3.1 Comparable corpora

The French and Japanese documents were harvested from the Web by native speakers of each language who are not domain specialists. The texts are from the medical domain, within the sub-domain of diabetes and nutrition. Document harvesting was carried out by a domain-based search, then by manual selection. The search for documents sharing the same domain can be achieved using keywords reflecting the specialized domain: for French, *diabète and obésité* (diabetes and obesity); for Japanese, 糖尿病 and 肥満. Then the documents were classified according to the type of discourse: scientific or popularized science. At present, the selection and classification phases are carried out manually although

[7] the French word order is inverted to take into account the different constraints between French and Japanese.

research into how to automatize these two steps is ongoing. Table 2 shows the main features of the harvested comparable corpora: the number of documents, and the number of words for each language and each type of discourse.

	French		Japanese	
	doc.	words	doc.	words
Scientific	65	425,781	119	234,857
Popular science	183	267,885	419	572,430
Total	248	693,666	538	807,287

Table 2: Comparable corpora statistics

From these documents, we created two comparable corpora:

- [scientific corpora], composed only of scientific documents;
- [mixed corpora], composed of both popular and scientific documents.

3.2 Bilingual dictionary

The French-Japanese bilingual dictionary required for the translation phase is composed of four dictionaries freely available from the Web[8], and of the French-Japanese Scientific Dictionary (1989). It contains about 173,156 entries (114,461 single words and 58,695 multi words) with an average of 2.1 translations per entry.

3.3 Terminology reference lists

To evaluate the quality of the terminology mining chain, we built two bilingual terminology reference lists which include either SWTs or SMTs and MWTs:

- [lexicon 1] 100 French SWTs of which the translation are Japanese SWTs.
- [lexicon 2] 60 French SWTs and MWTs of which the translation could be Japanese SWTs or MWTs.

These lexicons contains terms that occur at least twice in the scientific corpus, have been identified monolingually by both the French and the Japanese term extraction programs, and are found in either the UMLS[9] thesaurus or in the French part of the *Grand dictionnaire terminologique*[10] in the domain of medicine. These constraints prevented us from obtaining 100 French SWTs and MWTs for lexicon 2. The main reasons for this are the small number of UMLS terms dealing with the sub-domain of diabetes and the great difference between the linguistic structures of French and Japanese terms: French pattern definitions tend to cover more phrasal units while Japanese pattern definitions focus more narrowly on compounds. So, even if monolingually the same percentage of terms are detected in both languages, this does not guarantee a good result in bilingual terminology extraction. For example, the French term *diabète de type 1* (Diabetes mellitus type I) extracted by the French term extraction program and found in UMLS was not extracted by the Japanese term extraction program although it appears frequently in the Japanese corpus (一 型 糖 尿 病).

In bilingual terminology mining from specialized comparable corpora, the terminology reference lists are often composed of a hundred words (180 SWTs in (Déjean and Gaussier, 2002) and 97 SWTs in (Chiao and Zweigenbaum, 2002)).

4 Experiments

In order to evaluate the influence of discourse type on the quality of bilingual terminology extraction, two experiments were carried out. Since the main studies relating to bilingual lexicon extraction from comparable corpora concentrate on finding translation candidates for SWTs, we first perform an experiment using [lexicon 1], which is composed of SWTs. In order to evaluate the hypothesis of this study, we then conducted a second experiment using [lexicon 2], which is composed of MWTs.

4.1 Alignment results for [lexicon 1]

Table 3 shows the results obtained. The first three columns indicate the number of translations found

[8] http://kanji.free.fr/, http://quebec-japon.com/lexique/index.php?a=index&d=25, http://dico.fj.free.fr/index.php, http://quebec-japon.com/lexique/index.php?a=index&d=3

[9] http://www.nlm.nih.gov/research/umls

[10] http://www.granddictionnaire.com/

	NB_{trans}	AVG_{pos}	$STDDEV_{pos}$	TOP_{10}	TOP_{20}
[scientific corpora]	64	11.6	20.2	49	52
[mixed corpora]	76	11.5	16.3	51	60

Table 3: Bilingual terminology extraction results for [lexicon 1]

	NB_{trans}	AVG_{pos}	$STDDEV_{pos}$	TOP_{10}	TOP_{20}
[scientific corpora]	32	16.1	21.9	18	25
[mixed corpora]	32	23.9	27.6	17	20

Table 4: Bilingual terminology extraction results for [lexicon 2]

(NB_{trans}), and the average (AVG_{pos}) and standard deviation ($STDDEV_{pos}$) positions for the translations in the ranked list of candidate translations. The other two columns indicate the percentage of French terms for which the correct translation was obtained among the top ten and top twenty candidates (TOP_{10}, TOP_{20}).

The results of this experiment (see Table 3) show that the terms belonging to [lexicon 1] were more easily identified in the corpus of scientific and popular documents (51% and 60% respectively for TOP_{10} and TOP_{20}) than in the corpus of scientific documents (49% and 52%). Since [lexicon 1] is composed of SWTs, these terms are not more characteristic of popular discourse than scientific discourse.

The frequency of the terms to be translated is an important factor in the vectorial approach. In fact, the higher the frequency of the term to be translated, the more the associated context vector will be discriminant. Table 5 confirms this hypothesis since the most frequent terms, such as *insuline* (#occ. 364 - *insulin*: インスリン), *obésité* (#occ. 333 - *obesity*: 肥満), and *prévention* (#occ. 120 - *prevention*: 予防), were the best translated.

	[2,10]	**[11,50]**	**[51,100]**	**[101,...]**
fr	3/17	12/29	17/23	28/31
jp	4/26	32/41	14/20	10/13

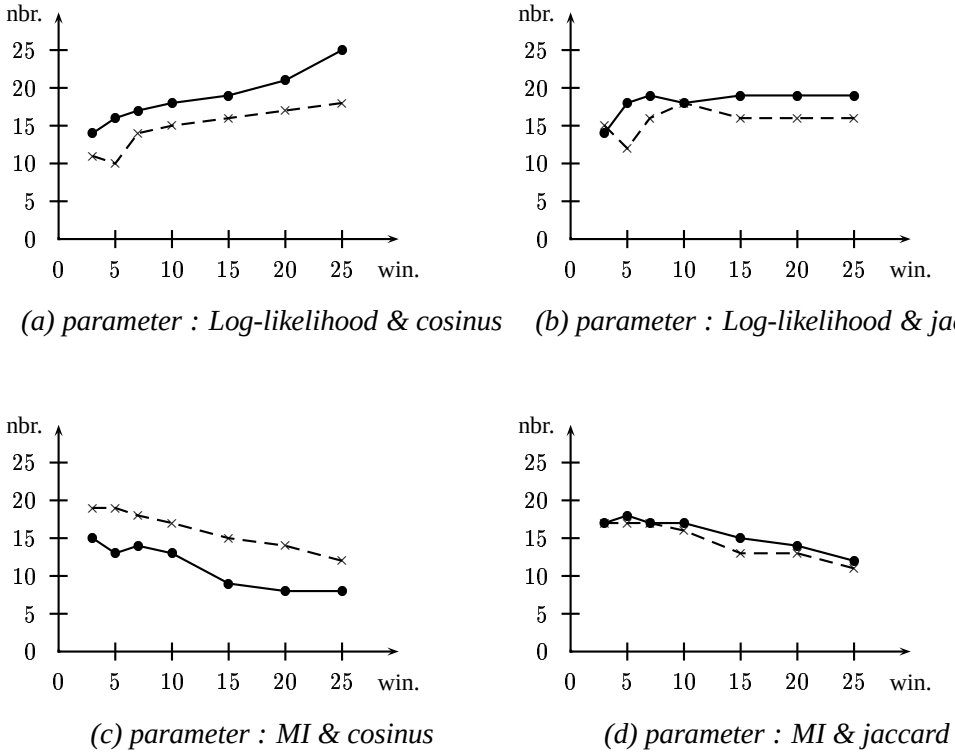
Table 5: Frequency in [corpus 2] of the terms translated belonging to [lexicon 1] (for TOP_{20})

As a baseline, (Déjean et al., 2002) obtain 43% and 51% for the first 10 and 20 candidates respectively in a 100,000-word medical corpus, and 79% and 84% in a multi-domain 8 million-word corpus. For single-item French-English words applied on a medical corpus of 0.66 million words, (Chiao and Zweigenbaum, 2002) obtained 61% and 94% precision on the top-10 and top-20 candidates. In our case, we obtained 51% and 60% precision for the top 10 and 20 candidates in a 1.5 million-word French/Japanese corpus.

4.2 Alignment results for [lexicon 2]

The analysis results in table 4 indicate only a small number of the terms in [lexicon 2] were found. Since we work with small-size corpora, this result is not surprising. Because multi-word terms are more specific than single-word terms, they tend to occur less frequently in a corpus and are more difficult to translate. Here, the terms belonging [lexicon 2] were more accurately identified from the corpus which consists of scientific documents than the corpus which consists of scientific and popular documents. In this instance, we obtained 30% and 42% precision for the top 10 and top 20 candidates in a 0.84 million-word scientific corpus. Moreover, if we count the number of terms which are correctly translated between [scientific corpora] and [mixed corpora], we find the majority of the translated terms with [mixed corpora] in those obtained with [scientific corpora][11] By combining parameters

[11] Here, $TOP_{10}18 \cap 17 = 15$, $TOP_{20}25 \cap 20 = 18$ and $NB_{trad}32 \cap 32 = 28$.



(a) parameter : Log-likelihood & cosinus

(b) parameter : Log-likelihood & jaccard

(c) parameter : MI & cosinus

(d) parameter : MI & jaccard

Figure 2: Evolution of the number of translations found in TOP_{20} according to the size of the contextual window for several combinations of parameters with [lexicon 2] ([scientific corpora] ——; [mixed corpora] - - -, the points indicated are the computed values)

such as the window size of the context vector, association score, and vector distance measure, the terms were often identified with more precision from the corpus consisting of scientific documents than the corpus consisting of scientific and popular documents (see Figure 2).

Here again, the most frequent terms (see Table 6), such as *diabète* (#occ. 899 - *diabetes*: 糖尿.病), *facteur de risque* (#occ. 267 - *risk factor*: 危険.因子), *hyperglycémie* (#occ. 127 - *hyperglycaemia*: 高.血糖), *tissu adipeux* (#occ. 62 - *adipose tissue*: 脂肪.組織) were the best translated. On the other hand, some terms with low frequency, such as *édulcorant* (#occ. 13 - *sweetener*: 甘味.料) and *choix alimentaire* (#occ. 11 - *feeding preferences*: 食品.選択), or very low frequency, such as *obésité massive* (#occ. 6 - *massive obesity*: 高度.肥満), were also identified with this approach.

	[2,10]	**[11,50]**	**[51,100]**	**[101,...]**
fr	1/11	11/25	6/14	7/10
jp	5/21	13/25	5/9	2/5

Table 6: Frequency in [scientific corpora] of translated terms belonging to [lexicon 2] (for TOP_{20})

5 Conclusion

This article describes a first attempt at compiling French-Japanese terminology from comparable corpora taking into account both single- and multi-word terms. Our claim was that a real comparability of the corpora is required to obtain relevant terms of the domain. This comparability should be based not only on the domain and the sub-domain but also on the type of discourse, which acts semantically upon the lexical units. The discourse categorization of documents allows lexical acquisition to increase pre-

cision despite the data sparsity problem that is often encountered for terminology mining and for language pairs not involving the English language, such as French-Japanese. We carried out experiments using two corpora of the specialised domain concerning diabetes and nutrition: one gathering documents from both scientific and popular science discourses, the other limited to scientific discourse. Our alignment results are close to previous works involving the English language, and are of better quality for the scientific corpus despite a corpus size that was reduced by half. The results demonstrate that the more frequent a term and its translation, the better the quality of the alignment will be, but also that the data sparsity problem could be partially solved by using comparable corpora of high quality.

References

Douglas Biber. 1994. Representativeness in corpus design. In A. Zampolli, N. Calzolari, and M. Palmer, editors, *Current Issues in Computational Linguistics: in Honour of Don Walker*, pages 377–407. Pisa: Giardini/Dordrecht: Kluwer.

Lynne Bowker and Jennifer Pearson. 2002. *Working with Specialized Language: A Practical Guide to Using Corpora*. London/New York: Routledge.

Yunbo Cao and Hang Li. 2002. Base Noun Phrase Translation Using Web Data and the EM Algorithm. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, pages 127–133, Tapei, Taiwan.

Yun-Chuang Chiao and Pierre Zweigenbaum. 2002. Looking for candidate translational equivalents in specialized, comparable corpora. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, pages 1208–1212, Tapei, Taiwan.

Béatrice Daille. 2003. Terminology Mining. In Maria Teresa Pazienza, editor, *Information Extraction in the Web Era*, pages 29–44. Springer.

Hervé Déjean and Éric Gaussier. 2002. Une nouvelle approche à l'extraction de lexiques bilingues à partir de corpus comparables. *Lexicometrica, Alignement lexical dans les corpus multilingues*, pages 1–22.

Hervé Déjean, Fatia Sadat, and Éric Gaussier. 2002. An approach based on multilingual thesauri and model combination for bilingual lexicon extraction. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, pages 218–224, Tapei, Taiwan.

French-Japanese Scientifi c Dictionary. 1989. Hakusuisha. 4th edition.

Pascale Fung. 1998. A Statistical View on Bilingual Lexicon Extraction: From Parallel Corpora to Nonparallel Corpora. In David Farwell, Laurie Gerber, and Eduard Hovy, editors, *Proceedings of the 3rd Conference of the Association for Machine Translation in the Americas (AMTA'98)*, pages 1–16, Langhorne, PA, USA. Springer.

Gregory Grefenstette. 1999. The Word Wide Web as a Resource for Example-Based Machine Translation Tasks. In *ASLIB'99 Translating and the Computer 21*, London, UK.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.

Carol Peters and Eugenio Picchi. 1998. Cross-language information retrieval: A system for comparable corpus querying. In Gregory Grefenstette, editor, *Cross-language information retrieval*, chapter 7, pages 81–90. Kluwer.

Reinhard Rapp. 1999. Automatic Identifi cation of Word Translations from Unrelated English and German Corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, pages 519–526, College Park, Maryland, USA.

Xavier Robitaille, Xavier Sasaki, Masatsugu Tonoike, Satoshi Sato, and Satoshi Utsuro. 2006. Compiling French-Japanese Terminologies from the Web. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL'06)*, pages 225–232, Trento, Italy.

Gerard Salton and Michael E. Lesk. 1968. Computer evaluation of indexing and text processing. *Journal of the Association for Computational Machinery*, 15(1):8–36.

Koichi Takeuchi, Kyo Kageura, Béatrice Daille, and Laurent Romary. 2004. Construction of grammar based term extraction model for japanese. In Sophia Ananadiou and Pierre Zweigenbaum, editors, *Proceeding of the COLING 2004, 3rd International Workshop on Computational Terminology (COMPUTERM'04)*, pages 91–94, Geneva, Switzerland.

T. T. Tanimoto. 1958. An elementary mathematical theory of classifi cation. Technical report, IBM Research.

Unsupervised Language Model Adaptation Incorporating Named Entity Information

Feifan Liu and Yang Liu
Department of Computer Science
The University of Texas at Dallas, Richardson, TX, USA
{ffliu,yangl}@hlt.utdallas.edu

Abstract

Language model (LM) adaptation is important for both speech and language processing. It is often achieved by combining a generic LM with a topic-specific model that is more relevant to the target document. Unlike previous work on unsupervised LM adaptation, this paper investigates how effectively using named entity (NE) information, instead of considering all the words, helps LM adaptation. We evaluate two latent topic analysis approaches in this paper, namely, clustering and Latent Dirichlet Allocation (LDA). In addition, a new dynamically adapted weighting scheme for topic mixture models is proposed based on LDA topic analysis. Our experimental results show that the NE-driven LM adaptation framework outperforms the baseline generic LM. The best result is obtained using the LDA-based approach by expanding the named entities with syntactically filtered words, together with using a large number of topics, which yields a perplexity reduction of 14.23% compared to the baseline generic LM.

1 Introduction

Language model (LM) adaptation plays an important role in speech recognition and many natural language processing tasks, such as machine translation and information retrieval. Statistical N-gram LMs have been widely used; however, they capture only local contextual information. In addition, even with the increasing amount of LM training data, there is often a mismatch problem because of differences in domain, topics, or styles. Adaptation of LM, therefore, is very important in order to better deal with a variety of topics and styles.

Many studies have been conducted for LM adaptation. One method is supervised LM adaptation, where topic information is typically available and a topic specific LM is interpolated with the generic LM (Kneser and Steinbiss, 1993; Suzuki and Gao, 2005). In contrast, various unsupervised approaches perform latent topic analysis for LM adaptation. To identify implicit topics from the unlabeled corpus, one simple technique is to group the documents into topic clusters by assigning only one topic label to a document (Iyer and Ostendorf, 1996). Recently several other methods in the line of latent semantic analysis have been proposed and used in LM adaptation, such as latent semantic analysis (LSA) (Bellegarda, 2000), probabilistic latent semantic analysis (PLSA) (Gildea and Hofmann, 1999), and LDA (Blei et al., 2003). Most of these existing approaches are based on the "bag of words" model to represent documents, where all the words are treated equally and no relation or association between words is considered.

Unlike prior work in LM adaptation, this paper investigates how to effectively leverage named entity information for latent topic analysis. Named entities are very common in domains such as newswire or broadcast news, and carry valuable information, which we hypothesize is topic indicative and useful for latent topic analysis. We compare different latent topic generation approaches as well as model adaptation methods, and propose an LDA based dynamic weighting method for the topic mixture model. Furthermore, we expand

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 672–679,
Prague, Czech Republic, June 2007.

named entities by incorporating other content words, in order to capture more topic information. Our experimental results show that the proposed method of incorporating named information in LM adaptation is effective. In addition, we find that for the LDA based adaptation scheme, adding more content words and increasing the number of topics can further improve the performance significantly.

The paper is organized as follows. In Section 2 we review some related work. Section 3 describes in detail our unsupervised LM adaptation approach using named entities. Experimental results are presented and discussed in Section 4. Conclusion and future work appear in Section 5.

2 Related Work

There has been a lot of previous related work on LM adaptation. Suzuki and Gao (2005) compared different supervised LM adaptation approaches, and showed that three discriminative methods significantly outperform the maximum a posteriori (MAP) method. For unsupervised LM adaptation, an earlier attempt is a cache-based model (Kuhn and Mori, 1990), developed based on the assumption that words appearing earlier in a document are likely to appear again. The cache concept has also been used to increase the probability of unseen but topically related words, for example, the trigger-based LM adaptation using the maximum entropy approach (Rosenfeld, 1996).

Latent topic analysis has recently been investigated extensively for language modeling. Iyer and Ostendorf (1996) used hard clustering to obtain topic clusters for LM adaptation, where a single topic is assigned to each document. Bellegarda (2000) employed Latent Semantic Analysis (LSA) to map documents into implicit topic sub-spaces and demonstrated significant reduction in perplexity and word error rate (WER). Its probabilistic extension, PLSA, is powerful for characterizing topics and documents in a probabilistic space and has been used in LM adaptation. For example, Gildea and Hofmann (1999) reported noticeable perplexity reduction via a dynamic combination of many unigram topic models with a generic trigram model. Proposed by Blei et al. (2003), Latent Dirichlet Allocation (LDA) loosens the constraint of the document-specific fixed weights by using a prior distribution and has quickly become one of the most popular probabilistic text modeling techniques. LDA can overcome the drawbacks in the PLSA model, and has been shown to outperform PLSA in corpus perplexity and text classification experiments (Blei et al., 2003). Tam and Schultz (2005) successfully applied the LDA model to unsupervised LM adaptation by interpolating the background LM with the dynamic unigram LM estimated by the LDA model. Hsu and Glass (2006) investigated using hidden Markov model with LDA to allow for both topic and style adaptation. Mrva and Woodland (2006) achieved WER reduction on broadcast conversation recognition using an LDA based adaptation approach that effectively combined the LMs trained from corpora with different styles: broadcast news and broadcast conversation data.

In this paper, we investigate unsupervised LM adaptation using clustering and LDA based topic analysis. Unlike the clustering based interpolation method as in (Iyer and Ostendorf, 1996), we explore different distance measure methods for topic analysis. Different from the LDA based framework as in (Tam and Schultz, 2005), we propose a novel dynamic weighting scheme for the topic adapted LM. More importantly, the focus of our work is to investigate the role of named entity information in LM adaptation, which to our knowledge has not been explored.

3 Unsupervised LM Adaptation Integrating Named Entities (NEs)

3.1 Overview of the NE-driven LM Adaptation Framework

Figure 1 shows our unsupervised LM adaptation framework using NEs. For training, we use the text collection to train the generic word-based N-gram LM. Then we apply named entity recognition (NER) and topic analysis to train multiple topic specific N-gram LMs. During testing, NER is performed on each test document, and then a dynamically adaptive LM based on the topic analysis result is combined with the general LM. Note that in this figure, we evaluate the performance of LM adaptation using the perplexity measure. We will evaluate this framework for N-best or lattice rescoring in speech recognition in the future.

In our experiments, different topic analysis methods combined with different topic matching and adaptive schemes result in several LM adapta-

tion paradigms, which are described below in details.

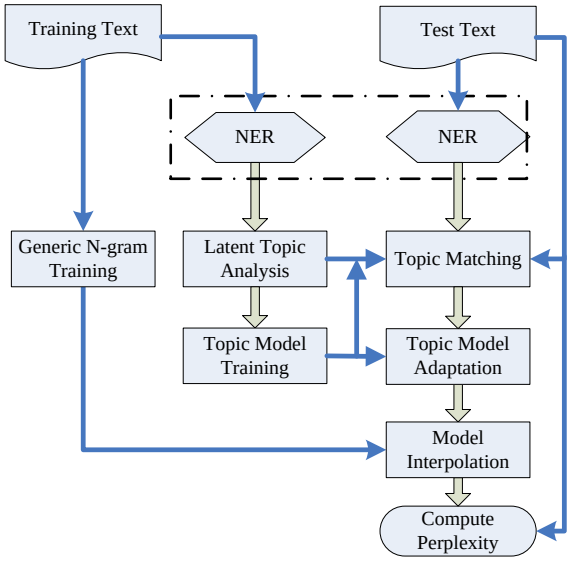


Figure 1. Framework of NE-driven LM adaptation.

3.2 NE-based Clustering for LM Adaptation

Clustering is a simple unsupervised topic analysis method. We use NEs to construct feature vectors for the documents, rather than considering all the words as in most previous work. We use the CLUTO[1] toolkit to perform clustering. It finds a predefined number of clusters based on a specific criterion, for which we chose the following function:

$$(S_1 S_2 \cdots S_k)^* = \arg\max \sum_{i=1}^{K} \sqrt{\sum_{v,u \in S_i} sim(v,u)}$$

where K is the desired number of clusters, S_i is the set of documents belonging to the i^{th} cluster, v and u represent two documents, and $sim(v, u)$ is the similarity between them. We use the cosine distance to measure the similarity between two documents:

$$sim(v,u) = \frac{\vec{v} \cdot \vec{u}}{\| \vec{v} \| \cdot \| \vec{u} \|} \quad (1)$$

where $\vec{v}$ and $\vec{u}$ are the feature vectors representing the two documents respectively, in our experiments composed of NEs. For clustering, the elements in every feature vector are scaled based on their term frequency and inverse document frequency, a concept widely used in information retrieval.

After clustering, we train an N-gram LM, called a topic LM, for each cluster using the documents in it.

During testing, we identify the 'topic' for the test document, and interpolate the topic specific LM with the background LM, that is, if the test document belongs to the cluster S^*, we can predict a word w_k in the document given the word's history h_k using the following equation:

$$p(w_k \mid h_k) = \lambda p_{General}(w_k \mid h_k) + (1-\lambda) p_{Topic-S^*}(w_k \mid h_k) \quad (2)$$

where λ is the interpolation weight.

We investigate two approaches to find the topic assignment S^* for a given test document.

(A) cross-entropy measure

For a test document $d=w_1,w_2,\ldots,w_n$ with a word distribution $p_d(w)$ and a cluster S with a topic LM $p_s(w)$, the cross entropy $CE(d, S)$ can be computed as:

$$CE(d,S) = H(p_d, p_s) = -\sum_{i=1}^{n} p_d(w_i) \log_2 (p_s(w_i))$$

From the information theoretic perspective, the cluster with the lower cross entropy value is expected to be more topically correlated to the test document. For each test document, we compute the cross entropy values according to different clusters, and select the cluster S^* that satisfies:

$$S^* = \arg\min_{1 \le i \le K} CE(d, S_i)$$

(B) cosine similarity

For each cluster, its centroid can be obtained by:

$$cv_i = \frac{1}{n_i} \sum_{k=1}^{n_i} u_{ik}$$

where u_{ik} is the vector for the k^{th} document in the i^{th} cluster, and n_i is the number of documents in the i^{th} cluster. The distance between the test document and a cluster can then be easily measured by the cosine similarity function as in Equation (1). Our goal here is to find the cluster S* which the test document is closest to, that is,

$$S^* = \arg\max_{1 \le i \le K} \frac{\vec{d} \cdot \overrightarrow{cv_i}}{\| \vec{d} \| \cdot \| \overrightarrow{cv_i} \|}$$

[1] Available at http://glaros.dtc.umn.edu/gkhome/views/cluto

where $\vec{d}$ is the feature vector for the test document.

3.3 NE-based LDA for LM Adaptation

LDA model (Blei et al., 2003) has been introduced as a new, semantically consistent generative model, which overcomes overfitting and the problem of generating new documents in PLSA. It is a three-level hierarchical Bayesian model. Based on the LDA model, a document *d* is generated as follows.

- Sample a vector of *K* topic mixture weights θ from a prior Dirichlet distribution with parameter α:

$$f(\theta;\alpha)=\prod_{k=1}^{K}\theta_k^{\alpha_k-1}$$

- For each word *w* in *d*, pick a topic *k* from the multinomial distribution θ.
- Pick a word *w* from the multinomial distribution $\beta_{w,k}$ given the k^{th} topic.

For a document $d=w_1,w_2,\ldots w_n$, the LDA model assigns it the following probability:

$$p(d)=\int_\theta\left(\prod_{i=1}^{n}\sum_{k=1}^{K}\beta_{w_i k}\cdot\theta_k\right)f(\theta;\alpha)d\theta$$

We use the MATLAB topic Toolbox 1.3 (Griffiths et al., 2004) in the training set to obtain the document-topic matrix, DP, and the word-topic matrix, WP. Note that here "words" correspond to the elements in the feature vector used to represent the document (e.g., NEs). In the DP matrix, an entry c_{ik} represents the counts of words in a document d_i that are from a topic z_k (k=1,2,…,K). In the WP matrix, an entry f_{jk} represents the frequency of a word w_j generated from a topic z_k (k=1,2,…,K) over the training set.

For training, we assign a topic z_i^* to a document d_i such that $z_i^*=\underset{1\le k\le K}{\arg\max}\, c_{ik}$. Based on the documents belonging to the different topics, *K* topic N-gram LMs are trained. This "hard clustering" strategy allows us to train an LM that accounts for all the words rather than simply those NEs used in LDA analysis, as well as use higher order N-gram LMs, unlike the 'unigram' based LDA in previous work.

For a test document $d = w_1,w_2,\ldots,w_n$ that is generated by multiple topics under the LDA assumption, we formulate a dynamically adapted topic model using the mixture of LMs from different topics:

$$p_{LDA-adapt}(w_k\mid h_k)=\sum_{i=1}^{K}\gamma_i\times p_{z_i}(w_k\mid h_k)$$

where $p_{z_i}(w_k\mid h_k)$ stands for the i^{th} topic LM, and γ_i is the mixture weight. Different from the idea of dynamic topic adaptation in (Tam and Schultz, 2005), we propose a new weighting scheme to calculate γ_i that directly uses the two resulting matrices from LDA analysis during training:

$$\gamma_k=\sum_{j=1}^{n}p(z_k\mid w_j)p(w_j\mid d)$$

$$p(z_k\mid w_j)=\frac{f_{jk}}{\sum_{p=1}^{K}f_{jp}},\quad p(w_j\mid d)=\frac{freq(w_j)}{\sum_{q=1}^{n}freq(w_q)}$$

where $freq(w_j)$ is the frequency of a word w_j in the document *d*. Other notations are consistent with the previous definitions.

Then we interpolate this adapted topic model with the generic LM, similar to Equation (2):

$$\begin{aligned}p(w_k\mid h_k)=&\lambda p_{General}(w_k\mid h_k)\\&+(1-\lambda)p_{LDA-adapt}(w_k\mid h_k)\end{aligned}\qquad(3)$$

4 Experiments

4.1 Experimental Setup

	# of files	# of words	# of NEs
Training Data	23,985	7,345,644	590,656
Test Data	2,661	831,283	65,867

Table 1. Statistics of our experimental data.

The data set we used is the LDC Mandarin TDT4 corpus, consisting of 337 broadcast news shows with transcriptions. These files were split into small pieces, which we call documents here, according to the topic segmentation information marked in the LDC's transcription. In total, there are 26,646 such documents in our data set. We randomly chose 2661 files as the test data (which is balanced for different news sources). The rest was used for topic analysis and also generic LM training. Punctuation marks were used to determine sentences in the transcriptions. We used the NYU NE tagger (Ji and Grishman, 2005) to recognize four kinds of NEs: Person, Location, Organi-

zation, and Geo-political. Table 1 shows the statistics of the data set in our experiments.

We trained trigram LMs using the SRILM toolkit (Stolcke, 2002). A fixed weight (i.e., λ in Equation (2) and (3)) was used for the entire test set when interpolating the generic LM with the adapted topic LM. Perplexity was used to measure the performance of different adapted LMs in our experiments.

4.2 Latent Topic Analysis Results

	Topic	# of Files	Top 10 Descriptive Items (Translated from Chinese)
Clustering Based	1	3526	U.S., Israel, Washington, Palestine, Bush, Clinton, Gore, Voice of America, Mid-East, Republican Party
	2	3067	Taiwan, Taipei, Mainland, Taipei City, Chinese People's Broadcasting Station, Shuibian Chen, the Executive Yuan, the Legislative Yuan, Democratic Progressive Party, Nationalist Party
	3	4857	Singapore, Japan, Hong Kong, Indonesia, Asia, Tokyo, Malaysia, Thailand, World, China
	4	4495	World, German, Landon, Russia, France, England, Xinhua News Agency, Europe, U.S., Italy
	5	7586	China, Beijing, Nation, China Central Television Station, Xinhua News Agency, Shanghai, World, State Council, Zemin Jiang, Beijing City
LDA Based	1	5859	China, Japan, Hong Kong, Beijing, Shanghai, World, Zemin Jiang, Macao, China Central Television Station, Africa
	2	3794	U.S., Bush, World, Gore, South Korea, North Korea, Clinton, George Walker Bush, Asia, Thailand
	3	4640	Singapore, Indonesia, Team, Israel, Europe, Germany, England, France, Palestine, Wahid
	4	4623	Taiwan, Russia, Mainland, India, Taipei, Shuibian Chen, Philippine, Estrada, Communist Party of China, RUS.
	5	4729	Xinhua News Agency, Nation, Beijing, World, Canada, Sydney, Brazil, Beijing City, Education Ministry, Cuba

Table 2. Topic analysis results using clustering and LDA (the number of documents and the top 10 words (NEs) in each cluster).

For latent topic analysis, we investigated two approaches using named entities, i.e., clustering and LDA. 5 latent topics were used in both approaches. Table 2 illustrates the resulting topics using the top 10 words in each topic. We can see that the words in the same cluster share some similarity and that the words in different clusters seem to be 'topically' different. Note that errors from automatic NE recognition may impact the clustering results. For example, '队/team' in the table (in topic 3 in LDA results) is an error and is less discriminative for topic analysis.

Table 3 shows the perplexity of the test set using the background LM (baseline) and each of the topic LMs, from clustering and LDA respectively. We can see that for the entire test set, a topic LM generally performs much worse than the generic LM. This is expected, since the size of a topic cluster is much smaller than that of the entire training set, and the test set may contain documents from different topics. However, we found that when using an optimal topic model (i.e., the topic LM that yields the lowest perplexity among the 5 topic LMs), 23.45% of the documents in the test set have a lower perplexity value than that obtained from the generic LM. This suggests that a topic model could benefit LM adaptation and motivates a dynamic topic adaptation approach for different test documents.

	Perplexity
Baseline	502.02
CL-1	1054.36
CL-2	1399.16
CL-3	919.237
CL-4	962.996
CL-5	981.072
LDA-1	1224.54
LDA-2	1375.97
LDA-3	1330.44
LDA-4	1328.81
LDA-5	1287.05

Table 3. Perplexity results using the baseline LM vs. the single topic LMs.

4.3 Clustering vs. LDA Based LM Adaptation

In this section, we compare three LM adaptation paradigms. As we discussed in Section 3, two of them are clustering based topic analysis, but using different strategies to choose the optimal cluster; and the third one is based on LDA analysis that

uses a dynamic weighting scheme for adapted topic mixture model.

Figure 2 shows the perplexity results using different interpolation parameters with the general LM. 5 topics were used in both clustering and LDA based approaches (as in Section 4.2). "CL-CE" means clustering based topic analysis via cross entropy criterion, "CL-Cos" represents clustering based topic analysis via cosine distance criterion, and "LDA-MIX" denotes LDA based topic mixture model, which uses 5 mixture topic LMs.

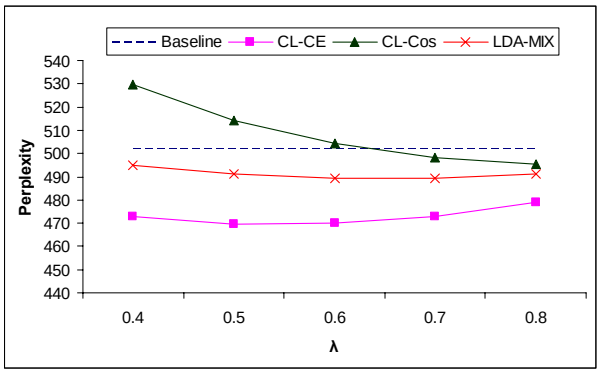


Figure 2. Perplexity using different LM adaptation approaches and different interpolation weights λ with the general LM.

We observe that all three adaptation approaches outperform the baseline when using a proper interpolation weight. "CL-CE" yields the best perplexity of 469.75 when λ is 0.5, a reduction of 6.46% against the baseline perplexity of 502.02. For clustering based adaptation, between the two strategies used to determine the topic for a test document, "CL-CE" outperforms "CL-Cos". This indicates that the cosine distance measure using only names is less effective than cross entropy for LM adaptation. In addition, cosine similarity does not match perplexity as well as the CE-based distance measure. Similarly, for the LDA based approach, using only NEs may not be sufficient to find appropriate weights for the topic model. This also explains the bigger interpolation weight for the general LM in CL-Cos and LDA-MIX than that in "CL-CE".

For a fair comparison between the clustering and LDA based LM adaptation approaches, we also evaluated using the topic mixture model for the clustering based approach and using only one topic in the LDA based method. For clustering based adaptation, we constructed topic mixture models using the weights obtained from a linear normalization of the two distance measures presented in Section 3.2. In order to use only one topic model in LDA based adaptation, we chose the topic cluster that has the largest weight in the adapted topic mixture model (as in Sec 3.3). Table 4 shows the perplexity for the three approaches (CL-Cos, CL-CE, and LDA) using the mixture topic models versus a single topic LM. We observe similar trends as in Figure 2 when changing the interpolation weight λ with the generic LM; therefore, in Table 4 we only present results for one optimal interpolation weight.

	Single-Topic	Mixture-Topic
CL-Cos (λ =0.7)	498.01	497.86
CL-CE (λ =0.5)	469.75	483.09
LDA (λ =0.7)	488.96	489.14

Table 4. Perplexity results using the adapted topic model (single vs. mixture) for clustering and LDA based approaches.

We can see from Table 4 that using the mixture model in clustering based adaptation does not improve performance. This may be attributed to how the interpolation weights are calculated. For example, only names are used in cosine distance, and the normalized distance may not be appropriate weights. We also notice negligible difference when only using one topic in the LDA based framework. This might be because of the small number of topics currently used. Intuitively, using a mixture model should yield better performance, since LDA itself is based on the assumption of generating words from multiple topics. We will investigate the impact of the number of topics on LM adaptation in Section 4.5.

4.4 Effect of Different Feature Configurations on LM Adaptation

We suspect that using only named entities may not provide enough information about the 'topics' of the documents, therefore we investigate expanding the feature vectors with other words. Since generally content words are more indicative of the topic of a document than function words, we used a POS tagger (Hillard et al., 2006) to select words for latent topic analysis. We kept words with three POS classes: noun (NN, NR, NT), verb (VV), and modi-

fier (JJ), selected from the LDC POS set[2]. This is similar to the removal of stop words widely used in information retrieval.

Figure 3 shows the perplexity results for three different feature configurations, namely, all-words (w), names (n), and names plus syntactically filtered items (n+), for the CL-CE and LDA based approaches. The LDA based LM adaptation paradigm supports our hypothesis. Using named information instead of all the words seems to efficiently eliminate redundant information and achieve better performance. In addition, expanding named entities with syntactically filtered items yields further improvement. For CL-CE, using named information achieves the best result among the three configurations. This might be because that the clustering method is less powerful in analyzing the principal components as well as dealing with redundant information than the LDA model.

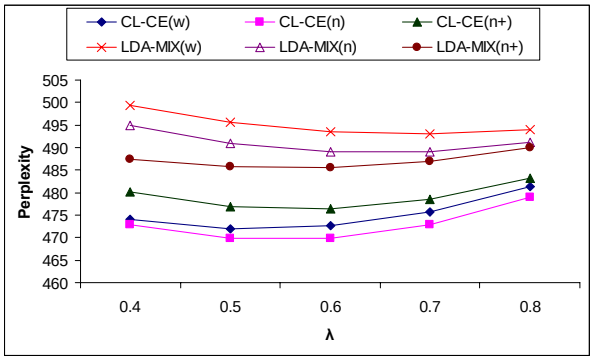


Figure 3. Comparison of perplexity using different feature configurations.

4.5 Impact of Predefined Topic Number on LM Adaptation

LDA based topic analysis typically uses a large number of topics to capture the fine grained topic space. In this section, we evaluate the effect of the number of topics on LM adaptation. For comparison, we evaluate this for both LDA and CL-CE, similar to Section 4.3. We use the "n+" feature configuration as in Section 4.4, that is, names plus POS filtered items. When using a single-topic adapted model in the LDA or CL-CE based approach, finer-grained topic analysis (i.e., increasing the number of topics) leads to worse performance mainly because of the smaller clusters for each topic; therefore, we only show results here using the mixture topic adapted models. Figure 4 shows the perplexity results using different numbers of topics. The interpolation weight λ with the general LM is 0.5 in all the experiments. For the topic mixture LMs, we used a maximum of 9 mixtures (a limitation in the current SRILM toolkit) when the number of topics is greater than 9.

We observe that as the number of topics increases, the perplexity reduces significantly for LDA. When the number of topics is 50, the adapted LM using LDA achieves a perplexity reduction of 11.35% compared to using 5 topics, and 14.23% against the baseline generic LM. Therefore, using finer-grained multiple topics in dynamic adaptation improves system performance. When the number of topics increases further, e.g., to 100, the performance degrades slightly. This might be due to the limitation of the number of the topic mixtures used. A similar trend is observable for the CL-CE approach, but the effect of the topic number is much greater in LDA than CL-CE.

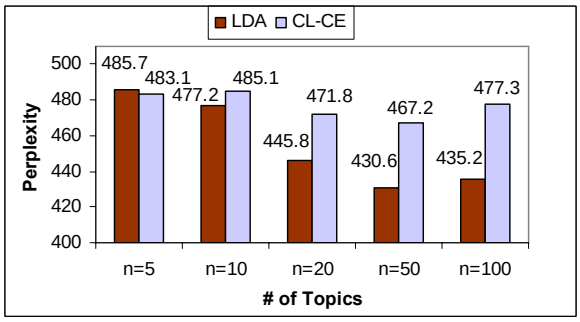


Figure 4. Perplexity results using different predefined numbers of topics for LDA and CL-CE.

4.6 Discussion

As we know, although there is an increasing amount of training data available for LM training, it is still only for limited domains and styles. Creating new training data for different domains is time consuming and labor intensive, therefore it is very important to develop algorithms for LM adaptation. We investigate leveraging named entities in the LM adaptation task. Though some errors of NER may be introduced, our experimental results have shown that exploring named information for topic analysis is promising for LM adaptation.

Furthermore, this framework may have other advantages. For speech recognition, using NEs for topic analysis can be less vulnerable to recognition

[2] See http://www.cis.upenn.edu/~chinese/posguide.3rd.ch.pdf

errors. For instance, we may add a simple module to compute the similarity between two NEs based on the word tokens or phonetics, and thus compensate the recognition errors inside NEs. Whereas, word-based models, such as the traditional cache LMs, may be more sensitive to recognition errors that are likely to have a negative impact on the prediction of the current word. From this point of view, our framework can potentially be more robust in the speech processing task. In addition, the number of NEs in a document is much smaller than that of the words, as shown in Table 1; hence, using NEs can also reduce the computational complexity, in particular in topic analysis for training.

5 Conclusion and Future Work

We compared several unsupervised LM adaptation methods leveraging named entities, and proposed a new dynamic weighting scheme for topic mixture model based on LDA topic analysis. Experimental results have shown that the NE-driven LM adaptation approach outperforms using all the words, and yields perplexity reduction compared to the baseline generic LM. In addition, we find that for the LDA based method, adding other content words, combined with an increased number of topics, can further improve the performance, achieving up to 14.23% perplexity reduction compared to the baseline LM.

The experiments in this paper combine models primarily through simple linear interpolation. Thus one direction of our future work is to develop algorithms to automatically learn appropriate interpolation weights. In addition, our work in this paper has only showed promising results in perplexity reduction. We will investigate using this framework of LM adaptation for N-best or lattice rescoring in speech recognition.

Acknowledgements

We thank Mari Ostendorf, Mei-Yuh Hwang, and Wen Wang for useful discussions, and Heng Ji for sharing the Mandarin named entity tagger. This work is supported by DARPA under Contract No. HR0011-06-C-0023. Any opinions expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

J. Bellegarda. 2000. Exploiting Latent Semantic Information in Statistical Language Modeling. In IEEE Transactions on Speech and Audio Processing. 88(80):1279-1296.

D. Blei, A. Ng, and M. Jordan. 2003. Latent Dirichlet Allocation. Journal of Machine Learning Research. 3:993-1022.

D. Gildea and T. Hofmann. 1999. Topic-Based Language Models using EM. In Proc. of Eurospeech.

T. Griffiths, M. Steyvers, D. Blei, and J. Tenenbaum. 2004. Integrating Topics and Syntax. Adv. in Neural Information Processing Systems. 17:537-544.

D. Hillard, Z. Huang, H. Ji, R. Grishman, D. Hakkani-Tur, M. Harper, M. Ostendorf, and W. Wang. 2006. Impact of Automatic Comma Prediction on POS/Name Tagging of Speech. In Proc. of the First Workshop on Spoken Language Technology (SLT).

P. Hsu and J. Glass. 2006. Style & Topic Language Model Adaptation using HMM-LDA. In Proc. of EMNLP, pp:373-381.

R. Iyer and M. Ostendorf. 1996. Modeling Long Distance Dependence in Language: Topic Mixtures vs. Dynamic Cache Models. In Proc. of ICSLP.

H. Ji and R. Grishman. 2005. Improving NameTagging by Reference Resolution and Relation Detection. In Proc. of ACL. pp: 411-418.

R. Kneser and V. Steinbiss. 1993. On the Dynamic Adaptation of Stochastic language models. In Proc. of ICASSP, Vol 2, pp: 586-589.

R. Kuhn and R.D. Mori. 1990. A Cache-Based Natural Language Model for Speech Recognition. In IEEE Transactions on Pattern Analysis and Machine Intelligence, 12: 570-583.

D. Mrva and P.C. Woodland. 2006. Unsupervised Language Model Adaptation for Mandarin Broadcast Conversation Transcription. In Proc. of INTERSPEECH, pp:2206-2209.

R. Rosenfeld. 1996. A Maximum Entropy Approach to Adaptive Statistical Language Modeling. Computer, Speech and Language, 10:187-228.

A. Stolcke. 2002. SRILM – An Extensible Language Modeling Toolkit. In Proc. of ICSLP.

H. Suzuki and J. Gao. 2005. A Comparative Study on Language Model Adaptation Techniques Using New Evaluation Metrics, In Proc. of HLT/EMNLP.

Y.C. Tam and T. Schultz. 2005. Dynamic Language Model Adaptation Using Variational Bayes Inference. In Proc. of INTERSPEECH, pp:5-8.

Coordinate Noun Phrase Disambiguation in a Generative Parsing Model

Deirdre Hogan*
Computer Science Department
Trinity College Dublin
Dublin 2, Ireland
dhogan@computing.dcu.ie

Abstract

In this paper we present methods for improving the disambiguation of noun phrase (NP) coordination within the framework of a lexicalised history-based parsing model. As well as reducing noise in the data, we look at modelling two main sources of information for disambiguation: symmetry in conjunct structure, and the dependency between conjunct lexical heads. Our changes to the baseline model result in an increase in NP coordination dependency *f*-score from 69.9% to 73.8%, which represents a relative reduction in *f*-score error of 13%.

1 Introduction

Coordination disambiguation is a relatively little studied area, yet the correct bracketing of coordination constructions is one of the most difficult problems for natural language parsers. In the Collins parser (Collins, 1999), for example, dependencies involving coordination achieve an *f*-score as low as 61.8%, by far the worst performance of all dependency types.

Take the phrase *busloads of executives and their wives* (taken from the WSJ treebank). The coordinating conjunction (CC) *and* and the noun phrase *their wives* could attach to the noun phrase *executives*, as illustrated in Tree 1, Figure 1. Alternatively, *their wives* could be incorrectly conjoined to the noun phrase *busloads of executives* as in Tree 2, Figure 1.

As with PP attachment, most previous attempts at tackling coordination as a subproblem of parsing have treated it as a separate task to parsing and it is not always obvious how to integrate the methods proposed for disambiguation into existing parsing models. We therefore approach coordination disambiguation, not as a separate task, but from within the framework of a generative parsing model.

As noun phrase coordination accounts for over 50% of coordination dependency error in our baseline model we focus on NP coordination. Using a model based on the generative parsing model of (Collins, 1999) Model 1, we attempt to improve the ability of the parsing model to make the correct coordination decisions. This is done in the context of parse reranking, where the *n*-best parses output from Bikel's parser (Bikel, 2004) are reranked according to a generative history-based model.

In Section 2 we summarise previous work on coordination disambiguation. There is often a considerable bias toward symmetry in the syntactic structure of two conjuncts and in Section 3 we introduce new parameter classes to allow the model to prefer symmetry in conjunct structure. Section 4 is concerned with modelling the dependency between conjunct head words and begins by looking at how the different handling of coordination in noun phrases and base noun phrases (NPB) affects coordination disambiguation.[1] We look at how we might improve the model's handling of coordinate head-head dependencies by altering the model so that a common

* Now at the National Centre for Language Technology, Dublin City University, Ireland.

[1] A base noun phrase, as defined in (Collins, 1999), is a noun phrase which does not directly dominate another noun phrase, unless that noun phrase is possessive.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 680–687,
Prague, Czech Republic, June 2007.

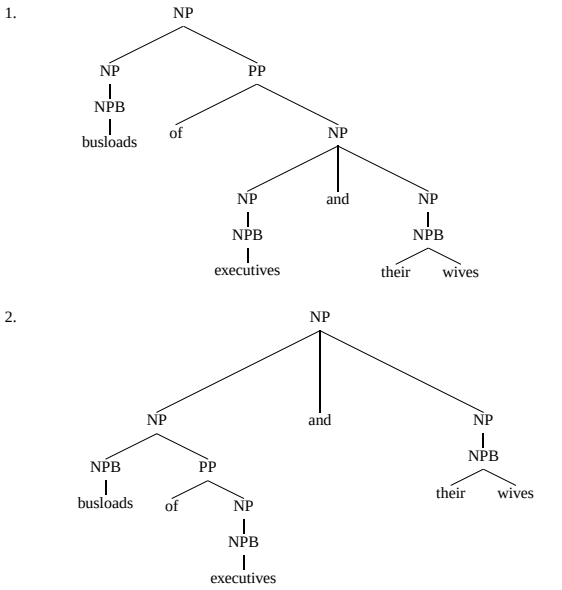


Figure 1: Tree 1. The correct noun phrase parse. Tree 2. The incorrect parse for the noun phrase.

parameter class is used for coordinate word probability estimation in both NPs and NPBs. In Section 4.2 we focus on improving the estimation of this parameter class by incorporating BNC data, and a measure of word similarity based on vector cosine similarity, to reduce data sparseness. In Section 5 we suggest a new head-finding rule for NPBs so that the lexicalisation process for coordinate NPBs is more similar to that of other NPs.

Section 6 examines inconsistencies in the annotation of coordinate NPs in the Penn Treebank which can lead to errors in coordination disambiguation. We show how some coordinate noun phrase inconsistencies can be automatically detected and cleaned from the data sets. Section 7 details how the model is evaluated, presents the experiments made and gives a breakdown of results.

2 Previous Work

Most previous attempts at tackling coordination have focused on a particular type of NP coordination to disambiguate. Both Resnik (1999) and Nakov and Hearst (2005) consider NP coordinations of the form *n1 and n2 n3* where two structural analyses are possible: *((n1 and n2) n3)* and *((n1) and (n2 n3))*. They aim to show more structure than is shown in trees following the Penn guidelines, whereas in our approach we aim to reproduce Penn guideline trees. To resolve the ambiguities, Resnik combines number agreement information of candidate conjoined nouns, an information theoretic measure of semantic similarity, and a measure of the appropriateness of noun-noun modification. Nakov and Hearst (2005) disambiguate by combining Web-based statistics on head word co-occurrences with other mainly heuristic information sources.

A probabilistic approach is presented in (Goldberg, 1999), where an unsupervised maximum entropy statistical model is used to disambiguate coordinate noun phrases of the form *n1 preposition n2 cc n3*. Here the problem is framed as an attachment decision: does *n3* attach 'high' to the first noun, *n1*, or 'low' to *n2*?

In (Agarwal and Boggess, 1992) the task is to identify pre-CC conjuncts which appear in text that has been part-of-speech (POS) tagged and semi-parsed, as well as tagged with semantic labels specific to the domain. The identification of the pre-CC conjunct is based on heuristics which choose the pre-CC conjunct that maximises the symmetry between pre- and post-CC conjuncts.

Insofar as we do not separate coordination disambiguation from the overall parsing task, our approach resembles the efforts to improve coordination disambiguation in (Kurohashi, 1994; Ratnaparkhi, 1994; Charniak and Johnson, 2005). In (Kurohashi, 1994) coordination disambiguation is carried out as the first component of a Japanese dependency parser using a technique which calculates similarity between series of words from the left and right of a conjunction. Similarity is measured based on matching POS tags, matching words and a thesaurus-based measure of semantic similarity. In both the discriminative reranker of Ratnaparkhi et al. (1994) and that of Charniak and Johnson (2005) features are included to capture syntactic parallelism across conjuncts at various depths.

3 Modelling Symmetry Between Conjuncts

There is often a considerable bias toward symmetry in the syntactic structure of two conjuncts, see for example (Dubey et al., 2005). Take Figure 2: If we take as level 0 the level in the coordinate sub-

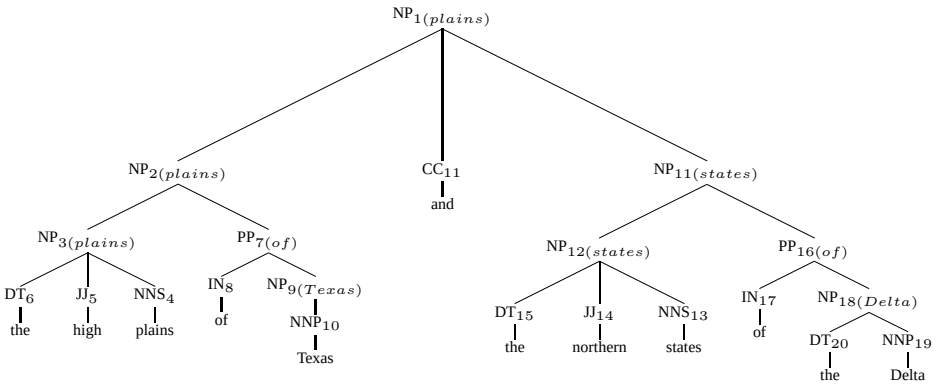


Figure 2: Example of symmetry in conjunct structure in a lexicalised subtree.

tree where the coordinating conjunction *CC* occurs, then there is exact symmetry in the two conjuncts in terms of non-terminal labels and head word part-of-speech tags for levels 0, 1 and 2. Learning a bias toward parallelism in conjuncts should improve the parsing model's ability to correctly attach a coordination conjunction and second conjunct to the correct position in the tree.

In history-based models, features are limited to being functions of the tree generated so far. The task is to incorporate a feature into the model that captures a particular bias yet still adheres to derivation-based restrictions. Parses are generated top-down, head-first, left-to-right. Each node in the tree in Figure 2 is annotated with the order the nodes are generated (we omit, for the sake of clarity, the generation of the *STOP* nodes). Note that when the decision to attach the second conjunct to the head conjunct is being made (i.e. Step 11, when the *CC* and *NP(states)* nodes are being generated) the subtree rooted at *NP(states)* has not yet been generated. Thus at the point that the conjunct attachment decision is made it is not possible to use information about symmetry of conjunct structure, as the structure of the second conjunct is not yet known.

It is possible, however, to condition on structure of the already generated head conjunct when building the internal structure of the second conjunct. In our model when the structure of the second conjunct is being generated we condition on features which are functions of the first conjunct. When generating a node N_i in the second conjunct, we retrieve the corresponding node $N_{i_{preCC}}$ in the first conjunct, via a left to right traversal of the first conjunct. For example, from Figure 2 the pre-*CC* node *NP(Texas)* is the node corresponding to *NP(Delta)* in the post-*CC* conjunct. From $N_{i_{preCC}}$ we extract information, such as its part-of-speech, for use as a feature when predicting a POS tag for the corresponding node in the post-*CC* conjunct.

When generating a second conjunct, instead of the usual parameter classes for estimating the probability of the head label C_h and the POS label of a dependent node t_i, we created two new parameter classes which are used only in the generation of second conjunct nodes:

$$P_{ccC_h}(C_h|\gamma(headC), C_p, w_p, t_p, t_{gp}, depth) \quad (1)$$

$$P_{cct_i}(t_i|\alpha(headC), dir, C_p, w_p, t_p, dist, t_{i\_1}, t_{i\_2}, depth) \quad (2)$$

where $\gamma(headC)$ returns the non-terminal label of $N_{i_{preCC}}$ for the node in question and $\alpha(headC)$ returns the POS tag of $N_{i_{preCC}}$. Both functions return *+NOMATCH+* if there is no $N_{i_{preCC}}$ for the node. Depth is the level of the post-*CC* conjunct node being generated.

4 Modelling Coordinate Head Words

Some noun pairs are more likely to be conjoined than others. Take again the trees in Figure 1. The two head nouns coordinated in Tree 1 are *executives* and *wives*, and in Tree 2: *busloads* and *wives*. Clearly, the former pair of head nouns is more likely and, for the purpose of discrimination, the model would benefit if it could learn that *executives and wives* is a more likely combination than *busloads and wives*.

Bilexical head-head dependencies of the type found in coordinate structures are a somewhat dif-

ferent class of dependency to modifier-head dependencies. In *the fat cat*, for example, there is clearly one head to the noun phrase: *cat*. In *cats and dogs* however there are two heads, though in the parsing model just one is chosen, somewhat arbitrarily, to head the entire noun phrase.

In the baseline model there is essentially one parameter class for the estimation of word probabilities:

$$P_{word}(w_i|H(i)) \quad (3)$$

where w_i is the lexical head of constituent i and $H(i)$ is the history of the constituent. The history is made up of conditioning features chosen from structure that has already been determined in the top-down derivation of the tree.

In Section 4.1 we discuss how though the coordinate head-head dependency is captured for NPs, it is not captured for NPBs. We look at how we might improve the model's handling of coordinate head-head dependencies by altering the model so that a common parameter class in (4) is used for coordinate word probability estimation in both NPs and NPBs.

$$P_{coordWord}(w_i|w_p, H(i)) \quad (4)$$

In Section 4.2 we focus on improving the estimation of this parameter class by reducing data sparseness.

4.1 Extending $P_{coordWord}$ to Coordinate NPBs

In the baseline model each node in the tree is annotated with a coordination flag which is set to true for the node immediately following the coordinating conjunction. For coordinate NPs the head-head dependency is captured when this flag is set to true. In Figure 1, discarding for simplicity the other features in the history, the probability of the coordinate head *wives*, is estimated in Tree 1 as:

$$P_{word}(w_i = wives|coord = true, w_p = executives, ...) \quad (5)$$

and in Tree 2:

$$P_{word}(w_i = wives|coord = true, w_p = busloads, ...) \quad (6)$$

where w_p is the head word of the node to which the node headed by w_i is attaching and $coord$ is the coordination flag.

Unlike NPs, in NPBs (i.e. flat, non-recursive NPs) the coordination flag is not used to mark whether a node is a coordinated head or not. This flag is always set to false for NPBs. In addition, modifiers within NPBs are conditioned on the previously generated modifier rather than the head of the phrase.[2] This means that in an NPB such as *(cats and dogs)*, the estimate for the word *cats* will look like:

$$P_{word}(w_i = cats|coord = false, w_p = and, ...) \quad (7)$$

In our new model, for NPs, when the coordination flag is set to true, we use the parameter class in (4) to estimate the probability of one lexical head noun, given another. For NPBs, if a noun is generated directly after a CC then it is taken to be a coordinate head, w_i, and conditioned on the noun generated before the coordinating conjunction, which is chosen as w_p, and also estimated using (4).

4.2 Estimating the $P_{coordWord}$ parameter class

Data for bilexical statistics are particularly sparse. In order to decrease the sparseness of the coordinate head noun data, we extracted from the BNC examples of coordinate head noun pairs. We extracted all noun pairs occurring in a pattern of the form: *noun cc noun*, as well as lists of any number of nouns separated by commas and ending in *cc noun*.[3] To this data we added all head noun pairs from the WSJ that occurred together in a coordinate noun phrase, identified when the coordination flag was set to true. Every occurrence n_i *CC* n_j was also counted as an occurrence of n_j *CC* n_i. This further helps reduce sparseness.

The probability of one noun, n_i being coordinated with another n_j can be calculated simply as:

$$P_{lex}(n_i|n_j) = \frac{|n_i n_j|}{|n_j|} \quad (8)$$

Again to reduce data sparseness, we introduce a measure of word similarity. A word can be represented as a vector where every dimension of the vector represents another word type. The values of the vector components, the term weights, are derived from word co-occurrence counts. Cosine similarity between two word vectors can then be used to measure the similarity of two words. Measures of

[2] A full explanation of the handling of coordination in the model is given in (Bikel, 2004).

[3] Extracting coordinate noun pairs from the BNC in such a fashion follows work on networks of concepts described in (Widdows, 2004).

similarity between words based on similarity of co-occurrence vectors have been used before, for example, for word sense disambiguation (Schütze, 1998) and for PP-attachment disambiguation (Zhao and Lin, 2004). Our measure resembles that of (Caraballo, 99) where co-occurrence is also defined with respect to coordination patterns, although the experimental details in terms of data collection and vector term weights differ.

We can now incorporate the similarity measure into the probability estimate of (8) to give a new *k*-NN style method of estimating bilexical statistics based on weighting events according to the word similarity measure:

$$P_{sim}(n_i|n_j) = \frac{\sum_{n_x \in N(n_j)} sim(n_j, n_x)|n_i n_x|}{\sum_{n_x \in N(n_j)} sim(n_j, n_x)|n_x|} \quad (9)$$

where $sim(n_j, n_x)$ is a similarity score between words n_j and n_x and $N(n_j)$ is the set of words in the neighbourhood of n_j. This neighbourhood can be based on the k-nearest neighbours of n_j, where nearness is measured with the similarity function.

In order to smooth the bilexical estimate in (9) we combine it with another estimate, trained from WSJ data, by way of linear interpolation:

$$P_{coordWord}(n_i|n_j) = \lambda_{n_j} P_{sim}(n_i|n_j) + (1 - \lambda_{n_j}) P_{MLE}(n_i|t_i) \quad (10)$$

where t_i is the POS tag of word n_i, $P_{MLE}(n_i|t_i)$ is the maximum-likelihood estimate calculated from annotated WSJ data, and λ_{n_j} is calculated as in (11). In (11) we adapt the Witten-Bell method for the calculation of the weight λ, as used in the Collins parser, so that it incorporates the similarity measure for all words in the neighbourhood of n_j.

$$\lambda_{n_j} = \frac{\sum_{n_x \in N(n_j)} sim(n_j, n_x)|n_x|}{\sum_{n_x \in N(n_j)} sim(n_j, n_x)(|n_x| + CD(n_x))} \quad (11)$$

where C is a constant that can be optimised using held-out data and $D(n_j)$ is the diversity of a word n_j: the number of distinct words with which n_j has been coordinated in the training set.

The estimate in (9) can be viewed as the estimate with the more general history context than that of (8) because the context includes not only n_j but also words similar to n_j. The final probability estimate for $P_{coordWord}$ is calculated as the most specific estimate, P_{lex}, combined via regular Witten-Bell interpolation with the estimate in (10).

5 NPB Head-Finding Rules

Head-finding rules for coordinate NPBs differ from coordinate NPs.[4] Take the following two versions of the noun phrase *hard work and harmony*: (c) *(NP (NPB hard work and harmony))* and (d) *(NP (NP (NPB hard work)) and (NP (NPB harmony)))*. In the first example, *harmony* is chosen as head word of the NP; in example (d) the head of the entire NP is *work*. The choice of head affects the various dependencies in the model. However, in the case of two coordinate NPs which, as in the above example, cover the same span of words and differ only in whether the coordinate noun phrase is flat as in (c) or structured as in (d), the choice of head for the phrase is not particularly informative. In both cases the head words being coordinated are the same and either word could plausibly head the phrase; discrimination between trees in such cases should not be influenced by the choice of head, but rather by other, salient features that distinguish the trees.[5]

We would like to alter the head-finding rules for coordinate NPBs so that, in cases like those above, the word chosen to head the entire coordinate noun phrase would be the same for both base and non-base noun phrases. We experiment with slightly modified head-finding rules for coordinate NPBs. In an NPB such as *NPB → n1 CC n2 n3*, the head rules remain unchanged and the head of the phrase is (usually) the rightmost noun in the phrase. Thus, when *n2* is immediately followed by another noun the default is to assume nominal modifier coordination and the head rules stay the same. The modification we make to the head rules for NPBs is as follows: when *n2* is *not* immediately followed by a noun then the noun chosen to head the entire phrase is *n1*.

6 Inconsistencies in WSJ Coordinate NP Annotation

An inspection of NP coordination error in the baseline model revealed inconsistencies in WSJ annota-

[4] See (Collins, 1999) for the rules used in the baseline model.

[5] For example, it would be better if discrimination was largely based on whether *hard* modifies both *work* and *harmony* (c), or whether it modifies *work* alone (d).

tion. In this section we outline some types of coordinate NP inconsistency and outline a method for detecting some of these inconsistencies, which we later use to automatically clean noise from the data. Eliminating noise from treebanks has been previously used successfully to increase overall parser accuracy (Dickinson and Meurers, 2005).

The annotation of NPs in the Penn Treebank (Bies et al., 1995) follows somewhat different guidelines to that of other syntactic categories. Because their interpretation is so ambiguous, no internal structure is shown for nominal modifiers. For NPs with more than one head noun, if the only unshared modifiers in the constituent are nominal modifiers, then a flat structure is also given. Thus in *(NP the Manhattan phone book and tour guide)*[6] a flat structure is given because although *the* is a non-nominal modifier, it is shared, modifying both *tour guide* and *phone book*, and all other modifiers in the phrase are nominal.

However, we found that out of 1,417 examples of NP coordination in sections 02 to 21, involving phrases containing only nouns (common nouns or a mixture of common and proper nouns) and the coordinating conjunction, as many as 21.3%, contrary to the guidelines, were given internal structure, instead of a flat annotation. When all proper nouns are involved this phenomenon is even more common.[7]

Another common source of inconsistency in coordinate noun phrase bracketing occurs when a non-nominal modifier appears in the coordinate noun phrase. As previously discussed, according to the guidelines the modifier is annotated flat if it is shared. When the non-nominal modifier is unshared, more internal structure is shown, as in: *(NP (NP (NNS fangs)) (CC and) (NP (JJ pointed) (NNS ears)))*. However, the following two structured phrases, for example, were given a completely flat structure in the treebank: (a) *(NP (NP (NN oversight))(CC and) (NP (JJ disciplinary)(NNS procedures)))*, (b) *(NP (ADJP (JJ moderate)(CC and)(JJ low-cost))(NN housing))*. If we follow the guidelines then any coordinate NPB which ends with the following tag sequence can be automatically detected as incorrectly bracketed: *CC/non-nominal modifier/noun*. This is because either the non-nominal modifier, which is unambiguously unshared, is part of a noun phrase as (a) above, or it conjoined with another modifier as in (b). We found 202 examples of this in the training set, out of a total of 4,895 coordinate base noun phrases.

Finally, inconsistencies in POS tagging can also lead to problems with coordination. Take the bigram *executive officer*. We found 151 examples in the training set of a base noun phrase which ended with this bigram. 48% of the cases were POS tagged *JJ NN*, 52% tagged *NN NN*. [8] This has repercussions for coordinate noun phrase structure, as the presence of an adjectival pre-modifier indicates a structured annotation should be given.

These inconsistencies pose problems both for training and testing. With a relatively large amount of noise in the training set the model learns to give structures, which should be very unlikely, too high a probability. In testing, given inconsistencies in the gold standard trees, it becomes more difficult to judge how well the model is doing. Although it would be difficult to automatically detect the POS tagging errors, the other inconsistencies outlined above can be detected automatically by simple pattern matching. Automatically eliminating such examples is a simple method of cleaning the data.

7 Experimental Evaluation

We use a parsing model similar to that described in (Hogan, 2005) which is based on (Collins, 1999) Model 1 and uses *k*-NN for parameter estimation. The *n*-best output from Bikel's parser (Bikel, 2004) is reranked according to this *k*-NN parsing model, which achieves an *f*-score of 89.4% on section 23. For the coordination experiments, sections 02 to 21 are used for training, section 23 for testing and the remaining sections for validation. Results are for sentences containing 40 words or less.

As outlined in Section 6, the treebank guidelines are somewhat ambiguous as to the appropriate bracketing for coordinate NPs which consist entirely of proper nouns. We therefore do not include, in the coordination test and validation sets, coordinate NPs where in the gold standard NP the leaf nodes consist entirely of proper nouns (or CCs or commas). In do-

[6] In this section we do not show the NPB levels.

[7] In the guidelines it is recognised however that proper names are frequently annotated with internal structure.

[8] According to the POS bracketing guidelines (Santorini, 1991) the correct sequence of POS tags should be *NN NN*.

ing so we hope to avoid a situation whereby the success of the model is measured in part by how well it can predict the often inconsistent bracketing decisions made for a particular portion of the treebank.

In addition, and for the same reasons, if a gold standard tree is inconsistent with the guidelines in either of the following two ways the tree is not used when calculating coordinate precision and recall of the model: the gold tree is a noun phrase which ends with the sequence *CC/non-nominal modifier/noun*; the gold tree is a structured coordinate noun phrase where each word in the noun phrase is a noun.[9] Call these inconsistencies type *a* and type *b* respectively. This left us with a coordination validation set consisting of 1064 coordinate noun phrases and a test set of 416 coordinate NPs from section 23.

A coordinate phrase was deemed correct if the parent constituent label, and the two conjunct node labels (at level 0) match those in the gold subtree and if, in addition, each of the conjunct head words are the same in both test and gold tree. This follows the definition of a coordinate dependency in (Collins, 1999). Based on these criteria, the baseline *f*-scores for test and validation set were 69.1% and 67.1% respectively. The coordination *f*-score for the oracle trees on section 23 is 83.56%. In other words: if an 'oracle' were to choose from each set of *n*-best trees the tree that maximised constituent precision and recall, then the resulting set of oracle trees would have a NP coordination dependency *f*-score of 83.56%. For the validation set the oracle trees coordination dependency *f*-score is 82.47%.

7.1 Experiments and Results

We first eliminated from the training set all coordinate noun phrase subtrees, of type *a* and type *b* described in Section 7. The effect of this on the validation set is outlined in Table 1, step 2.

For the new parameter class in (1) we found that the best results occurred when it was used only in conjuncts of depth 1 and 2, although the case base for this parameter class contained head events from all post-*CC* conjunct depths. Parameter class (2) was used for predicting POS tags at level 1 in right-of-head conjuncts, although again the sample contained events from all depths.

Model	*f*-score	significance
1. Baseline	67.1	
2. NoiseElimination	68.7	$\gg 1$
3. Symmetry	69.9	$> 2, \gg 1$
4. NPB head rule	70.6	NOT $> 3, > 2, \gg 1$
5. $P_{coordWord}$ WSJ	71.7	NOT $> 4, > 3, \gg 2$
6. BNC data	72.1	NOT $> 5, > 4, \gg 3$
7. $sim(w_i, w_p)$	72.4	NOT > 6, NOT $> 5, \gg 4$

Table 1: Results on the Validation Set. 1064 coordinate noun phrase dependencies. In the significance column $>$ means at level .05 and $\gg$ means at level .005, for McNemar's test of significance. Results are cumulative.

For the $P_{coordWord}$ parameter class we extracted 9961 coordinate noun pairs from the WSJ training set and 815,323 pairs from the BNC. As pairs are considered symmetric this resulted in a total of 1,650,568 coordinate noun events. The term weights for the word vectors were dampened co-occurrence counts, of the form: $1 + log(count)$. For the estimation of $P_{sim}(n_i|n_j)$ we found it too computationally expensive to calculate similarity measures between n_j and each word token collected. The best results were obtained when the neighbourhood of n_j was taken to be the k-nearest neighbours of n_j from among the set of word that had previously occurred in a coordination pattern with n_j, where *k* is 1000. Table 1 shows the effect of the $P_{coordWord}$ parameter class estimated from WSJ data only (step 5), with the addition of BNC data (step 6) and finally with the word similarity measure (step 7).

The result of these experiments, as well as that involving the change in the head-finding heuristics, outlined in Section 5, was an increase in coordinate noun phrase *f*-score from 69.9% to 73.8% on the test set. This represents a 13% relative reduction in coordinate *f*-score error over the baseline, and, using McNemar's test for significance, is significant at the 0.05 level ($p = 0.034$). The reranker *f*-score for all constituents (not excluding any coordinate NPs) for section 23 rose slightly from 89.4% to 89.6%, a small but significant increase in *f*-score.[10]

Finally, we report results on an unaltered coordination test set, that is, a test set from which no

[9] Recall from §6 that for this latter case the noun phrase should be flat - an NPB - rather than a noun phrase with internal structure.

[10] Significance was calculated using the software available at www.cis.upenn.edu/ dbikel/software.html.

noisy events were eliminated. The baseline coordination dependency *f*-score for all NP coordination dependencies (550 dependencies) from section 23 is 69.27%. This rises to 72.74% when all experiments described in Section 7 are applied, which is also a statistically significant increase ($p = 0.042$).

8 Conclusion and Future Work

This paper outlined a novel method for modelling symmetry in conjunct structure, for modelling the dependency between noun phrase conjunct head words and for incorporating a measure of word similarity in the estimation of a model parameter. We also demonstrated how simple pattern matching can be used to reduce noise in WSJ noun phrase coordination data. Combined, these techniques resulted in a statistically significant improvement in noun phrase coordination accuracy.

Coordination disambiguation necessitates information from a variety of sources. Another information source important to NP coordinate disambiguation is the dependency between non-nominal modifiers and nouns which cross CCs in NPBs. For example, modelling this type of dependency could help the model learn that the phrase *the cats and dogs* should be bracketed flat, whereas the phrase *the U.S. and Washington* should be given structure.

Acknowledgements We are grateful to the TCD Broad Curriculum Fellowship scheme and to the SFI Basic Research Grant 04/BR/CS370 for funding this research. Thanks to Pádraig Cunningham, Saturnino Luz, Jennifer Foster and Gerard Hogan for helpful discussions and feedback on this work.

References

Rajeev Agarwal and Lois Boggess. 1992. A Simple but Useful Approach to Conjunct Identification. In *Proceedings of the 30th ACL*.

Ann Bies, Mark Ferguson, Karen Katz and Robert MacIntyre. 1995. Bracketing Guidelines for Treebank II Style Penn Treebank Project. Technical Report. University of Pennsylvania.

Dan Bikel. 2004. *On The Parameter Space of Generative Lexicalized Statistical Parsing Models*. Ph.D. thesis, University of Pennsylvania.

Sharon Caraballo. 1999. Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th ACL*.

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

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Markus Dickinson and W. Detmar Meurers. 2005. Prune diseased branches to get healthy trees! How to find erroneous local trees in a treebank and why it matters. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT)*.

Amit Dubey, Patrick Sturt and Frank Keller. 2005. Parallelism in Coordination as an Instance of Syntactic Priming: Evidence from Corpus-based Modeling. In *Proceedings of the HLT/EMNP-05*.

Miriam Goldberg. 1999. An Unsupervised Model for Statistically Determining Coordinate Phrase Attachment. In *Proceedings of the 27th ACL*.

Deirdre Hogan. 2005. *k*-NN for Local Probability Estimation in Generative Parsing Models. In *Proceedings of the IWPT-05*.

Sadao Kurohashi and Makoto Nagao. 1994. A Syntactic Analysis Method of Long Japanese Sentences Based on the Detection of Conjunctive Structures. In *Computational Linguistics*, 20(4).

Preslav Nakov and Marti Hearst. 2005. Using the Web as an Implicit Training Set: Application to Structural Ambiguity Resolution. In *Proceedings of the HLT/EMNLP-05*.

Adwait Ratnaparkhi, Salim Roukos and R. Todd Ward. 1994. A Maximum Entropy Model for Parsing. In *Proceedings of the International Conference on Spoken Language Processing*.

Philip Resnik. 1999. Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. In *Journal of Artifi cial Intelligence Research*, 11:95-130, 1999.

Beatrice Santorini. 1991. Part-of-Speech Tagging Guidelines for the Penn Treebank Project. Technical Report. University of Pennsylvania.

Hinrich Schütze. 1998. Automatic Word Sense Discrimination. *Computational Linguistics*, 24(1):97-123.

Dominic Widdows. 2004. *Geometry and Meaning*. CSLI Publications, Stanford, USA.

Shaojun Zhao and Dekang Lin. 2004. A Nearest-Neighbor Method for Resolving PP-Attachment Ambiguity. In *Proceedings of the IJCNLP-04*.

A Unified Tagging Approach to Text Normalization

Conghui Zhu
Harbin Institute of Technology
Harbin, China
chzhu@mtlab.hit.edu.cn

Jie Tang
Department of Computer Science
Tsinghua University, China
jietang@tsinghua.edu.cn

Hang Li
Microsoft Research Asia
Beijing, China
hangli@microsoft.com

Hwee Tou Ng
Department of Computer Science
National University of Singapore, Singapore
nght@comp.nus.edu.sg

Tiejun Zhao
Harbin Institute of Technology
Harbin, China
tjzhao@mtlab.hit.edu.cn

Abstract

This paper addresses the issue of text normalization, an important yet often overlooked problem in natural language processing. By text normalization, we mean converting 'informally inputted' text into the canonical form, by eliminating 'noises' in the text *and* detecting paragraph and sentence boundaries in the text. Previously, text normalization issues were often undertaken in an ad-hoc fashion or studied separately. This paper first gives a formalization of the entire problem. It then proposes a unified tagging approach to perform the task using Conditional Random Fields (CRF). The paper shows that with the introduction of a small set of tags, most of the text normalization tasks can be performed within the approach. The accuracy of the proposed method is high, because the subtasks of normalization are interdependent and should be performed together. Experimental results on email data cleaning show that the proposed method significantly outperforms the approach of using cascaded models and that of employing independent models.

1 Introduction

More and more 'informally inputted' text data becomes available to natural language processing, such as raw text data in emails, newsgroups, forums, and blogs. Consequently, how to effectively process the data and make it suitable for natural language processing becomes a challenging issue. This is because informally inputted text data is usually very noisy and is not properly segmented. For example, it may contain extra line breaks, extra spaces, and extra punctuation marks; and it may contain words badly cased. Moreover, the boundaries between paragraphs and the boundaries between sentences are not clear.

We have examined 5,000 randomly collected emails and found that 98.4% of the emails contain noises (based on the definition in Section 5.1).

In order to perform high quality natural language processing, it is necessary to perform 'normalization' on informally inputted data first, specifically, to remove extra line breaks, segment the text into paragraphs, add missing spaces and missing punctuation marks, eliminate extra spaces and extra punctuation marks, delete unnecessary tokens, correct misused punctuation marks, restore badly cased words, correct misspelled words, and identify sentence boundaries.

Traditionally, text normalization is viewed as an engineering issue and is conducted in a more or less ad-hoc manner. For example, it is done by using rules or machine learning models at different levels. In natural language processing, several issues of text normalization were studied, but were only done separately.

This paper aims to conduct a thorough investigation on the issue. First, it gives a formalization of

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 688–695,
Prague, Czech Republic, June 2007.

the problem; specifically, it defines the subtasks of the problem. Next, it proposes a unified approach to the whole task on the basis of tagging. Specifically, it takes the problem as that of assigning tags to the input texts, with a tag representing deletion, preservation, or replacement of a token. As the tagging model, it employs Conditional Random Fields (CRF). The unified model can achieve better performances in text normalization, because the subtasks of text normalization are often interdependent. Furthermore, there is no need to define specialized models and features to conduct different types of cleaning; all the cleaning processes have been formalized and conducted as assignments of the three types of tags.

Experimental results indicate that our method significantly outperforms the methods using cascaded models or independent models on normalization. Our experiments also indicate that with the use of the tags defined, we can conduct most of the text normalization in the unified framework.

Our contributions in this paper include: (a) formalization of the text normalization problem, (b) proposal of a unified tagging approach, and (c) empirical verification of the effectiveness of the proposed approach.

The rest of the paper is organized as follows. In Section 2, we introduce related work. In Section 3, we formalize the text normalization problem. In Section 4, we explain our approach to the problem and in Section 5 we give the experimental results. We conclude the paper in Section 6.

2 Related Work

Text normalization is usually viewed as an engineering issue and is addressed in an ad-hoc manner. Much of the previous work focuses on processing texts in clean form, not texts in informal form. Also, prior work mostly focuses on processing one type or a small number of types of errors, whereas this paper deals with many different types of errors.

Clark (2003) has investigated the problem of preprocessing noisy texts for natural language processing. He proposes identifying token boundaries and sentence boundaries, restoring cases of words, and correcting misspelled words by using a source channel model.

Minkov et al. (2005) have investigated the problem of named entity recognition in informally inputted texts. They propose improving the performance of personal name recognition in emails using two machine-learning based methods: Conditional Random Fields and Perceptron for learning HMMs. See also (Carvalho and Cohen, 2004).

Tang et al. (2005) propose a cascaded approach for email data cleaning by employing Support Vector Machines and rules. Their method can detect email headers, signatures, program codes, and extra line breaks in emails. See also (Wong et al., 2007).

Palmer and Hearst (1997) propose using a Neural Network model to determine whether a period in a sentence is the ending mark of the sentence, an abbreviation, or both. See also (Mikheev, 2000; Mikheev, 2002).

Lita et al. (2003) propose employing a language modeling approach to address the case restoration problem. They define four classes for word casing: all letters in lower case, first letter in uppercase, all letters in upper case, and mixed case, and formalize the problem as assigning class labels to words in natural language texts. Mikheev (2002) proposes using not only local information but also global information in a document in case restoration.

Spelling error correction can be formalized as a classification problem. Golding and Roth (1996) propose using the Winnow algorithm to address the issue. The problem can also be formalized as that of data conversion using the source channel model. The source model can be built as an n-gram language model and the channel model can be constructed with confusing words measured by edit distance. Brill and Moore, Church and Gale, and Mayes et al. have developed different techniques for confusing words calculation (Brill and Moore, 2000; Church and Gale, 1991; Mays et al., 1991).

Sproat et al. (1999) have investigated normalization of non-standard words in texts, including numbers, abbreviations, dates, currency amounts, and acronyms. They propose a taxonomy of non-standard words and apply n-gram language models, decision trees, and weighted finite-state transducers to the normalization.

3 Text Normalization

In this paper we define text normalization at three levels: paragraph, sentence, and word level. The subtasks at each level are listed in Table 1. For example, at the paragraph level, there are two sub-

tasks: extra line-break deletion and paragraph boundary detection. Similarly, there are six (three) subtasks at the sentence (word) level, as shown in Table 1. Unnecessary token deletion refers to deletion of tokens like '-----' and '====', which are not needed in natural language processing. Note that most of the subtasks conduct 'cleaning' of noises, except paragraph boundary detection and sentence boundary detection.

Level	Task	Percentages of Noises
Paragraph	Extra line break deletion	49.53
	Paragraph boundary detection	
Sentence	Extra space deletion	15.58
	Extra punctuation mark deletion	0.71
	Missing space insertion	1.55
	Missing punctuation mark insertion	3.85
	Misused punctuation mark correction	0.64
	Sentence boundary detection	
Word	Case restoration	15.04
	Unnecessary token deletion	9.69
	Misspelled word correction	3.41

Table 1. Text Normalization Subtasks

As a result of text normalization, a text is segmented into paragraphs; each paragraph is segmented into sentences with clear boundaries; and each word is converted into the canonical form. After normalization, most of the natural language processing tasks can be performed, for example, part-of-speech tagging and parsing.

We have manually cleaned up some email data (cf., Section 5) and found that nearly all the noises can be eliminated by performing the subtasks defined above. Table 1 gives the statistics.

1. i'm thinking about buying a pocket
2. pc device for my wife this christmas,.
3. the worry that i have is that she won't
4. be able to sync it to her outlook express
5. contacts…

Figure 1. An example of informal text

I'm thinking about buying a Pocket PC device for my wife this Christmas.// The worry that I have is that she won't be able to sync it to her Outlook Express contacts.//

Figure 2. Normalized text

Figure 1 shows an example of informally inputted text data. It includes many typical noises. From line 1 to line 4, there are four extra line breaks at the end of each line. In line 2, there is an extra comma after the word 'Christmas'. The first word in each sentence and the proper nouns (e.g., 'Pocket PC' and 'Outlook Express') should be capitalized. The extra spaces between the words 'PC' and 'device' should be removed. At the end of line 2, the line break should be removed and a space is needed after the period. The text should be segmented into two sentences.

Figure 2 shows an ideal output of text normalization on the input text in Figure 1. All the noises in Figure 1 have been cleaned and paragraph and sentence endings have been identified.

We must note that dependencies (sometimes even strong dependencies) exist between different types of noises. For example, word case restoration needs help from sentence boundary detection, and vice versa. An ideal normalization method should consider processing all the tasks together.

4 A Unified Tagging Approach

4.1 Process

In this paper, we formalize text normalization as a tagging problem and employ a unified approach to perform the task (no matter whether the processing is at paragraph level, sentence level, or word level).

There are two steps in the method: preprocessing and tagging. In preprocessing, (A) we separate the text into paragraphs (i.e., sequences of tokens), (B) we determine tokens in the paragraphs, and (C) we assign possible tags to each token. The tokens form the basic units and the paragraphs form the sequences of units in the tagging problem. In tagging, given a sequence of units, we determine the most likely corresponding sequence of tags by using a trained tagging model. In this paper, as the tagging model, we make use of CRF.

Next we describe the steps (A)-(C) in detail and explain why our method can accomplish many of the normalization subtasks in Table 1.

(A). We separate the text into paragraphs by taking two or more consecutive line breaks as the endings of paragraphs.

(B). We identify tokens by using heuristics. There are five types of tokens: 'standard word', 'non-standard word', punctuation mark, space, and line break. Standard words are words in natural language. Non-standard words include several general 'special words' (Sproat et al., 1999), email address, IP address, URL, date, number, money, percentage, unnecessary tokens (e.g., '===' and

'###'), etc. We identify non-standard words by using regular expressions. Punctuation marks include period, question mark, and exclamation mark. Words and punctuation marks are separated into different tokens if they are joined together. Natural spaces and line breaks are also regarded as tokens.

(C). We assign tags to each token based on the type of the token. Table 2 summarizes the types of tags defined.

Token Type	Tag	Description
Line break	PRV	Preserve line break
	RPA	Replace line break by space
	DEL	Delete line break
Space	PRV	Preserve space
	DEL	Delete space
Punctuation mark	PSB	Preserve punctuation mark and view it as sentence ending
	PRV	Preserve punctuation mark without viewing it as sentence ending
	DEL	Delete punctuation mark
Word	AUC	Make all characters in uppercase
	ALC	Make all characters in lowercase
	FUC	Make the first character in uppercase
	AMC	Make characters in mixed case
Special token	PRV	Preserve the special token
	DEL	Delete the special token

Table 2. Types of tags

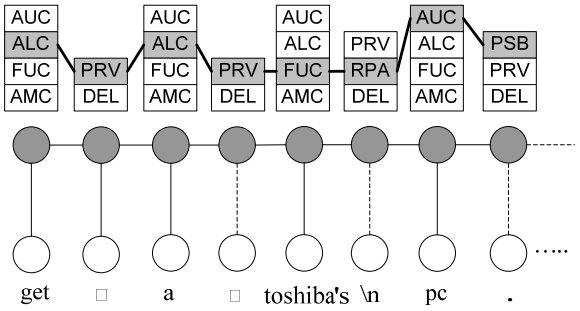


Figure 3. An example of tagging

Figure 3 shows an example of the tagging process. (The symbol '□' indicates a space). In the figure, a white circle denotes a token and a gray circle denotes a tag. Each token can be assigned several possible tags.

Using the tags, we can perform most of the text normalization processing (conducting seven types of subtasks defined in Table 1 and cleaning 90.55% of the noises).

In this paper, we do not conduct three subtasks, although we could do them in principle. These include missing space insertion, missing punctuation mark insertion, and misspelled word correction. In our email data, it corresponds to 8.81% of the noises. Adding tags for insertions would increase the search space dramatically. We did not do that due to computation consideration. Misspelled word correction can be done in the same framework easily. We did not do that in this work, because the percentage of misspelling in the data is small.

We do not conduct misused punctuation mark correction as well (e.g., correcting '.' with '?'). It consists of 0.64% of the noises in the email data. To handle it, one might need to parse the sentences.

4.2 CRF Model

We employ Conditional Random Fields (CRF) as the tagging model. CRF is a conditional probability distribution of a sequence of tags given a sequence of tokens, represented as $P(Y|X)$, where X denotes the token sequence and Y the tag sequence (Lafferty et al., 2001).

In tagging, the CRF model is used to find the sequence of tags Y^* having the highest likelihood $Y^* = \max_Y P(Y|X)$, with an efficient algorithm (the Viterbi algorithm).

In training, the CRF model is built with labeled data and by means of an iterative algorithm based on Maximum Likelihood Estimation.

Transition Features	
$y_{i-1}=y', y_i=y$	
$y_{i-1}=y', y_i=y, w_i=w$	
$y_{i-1}=y', y_i=y, t_i=t$	
State Features	
$w_i=w, y_i=y$	$t_i=t, y_i=y$
$w_{i-1}=w, y_i=y$	$t_{i-1}=t, y_i=y$
$w_{i-2}=w, y_i=y$	$t_{i-2}=t, y_i=y$
$w_{i-3}=w, y_i=y$	$t_{i-3}=t, y_i=y$
$w_{i-4}=w, y_i=y$	$t_{i-4}=t, y_i=y$
$w_{i+1}=w, y_i=y$	$t_{i+1}=t, y_i=y$
$w_{i+2}=w, y_i=y$	$t_{i+2}=t, y_i=y$
$w_{i+3}=w, y_i=y$	$t_{i+3}=t, y_i=y$
$w_{i+4}=w, y_i=y$	$t_{i+4}=t, y_i=y$
$w_{i-1}=w', w_i=w, y_i=y$	$t_{i-2}=t'', t_{i-1}=t', y_i=y$
$w_{i+1}=w', w_i=w, y_i=y$	$t_{i-1}=t', t_i=t, y_i=y$
	$t_i=t, t_{i+1}=t', y_i=y$
	$t_{i+1}=t', t_{i+2}=t'', y_i=y$
	$t_{i-2}=t'', t_{i-1}=t', t_i=t, y_i=y$
	$t_{i-1}=t'', t_i=t, t_{i+1}=t', y_i=y$
	$t_i=t, t_{i+1}=t', t_{i+2}=t'', y_i=y$

Table 3. Features used in the unified CRF model

4.3 Features

Two sets of features are defined in the CRF model: transition features and state features. Table 3 shows the features used in the model.

Suppose that at position i in token sequence x, w_i is the token, t_i the type of token (see Table 2), and y_i the possible tag. Binary features are defined as described in Table 3. For example, the transition feature $y_{i-1}=y'$, $y_i=y$ implies that if the current tag is y and the previous tag is y', then the feature value is true; otherwise false. The state feature $w_i=w$, $y_i=y$ implies that if the current token is w and the current label is y, then the feature value is true; otherwise false. In our experiments, an actual feature might be the word at position 5 is 'PC' and the current tag is AUC. In total, 4,168,723 features were used in our experiments.

4.4 Baseline Methods

We can consider two baseline methods based on previous work, namely cascaded and independent approaches. The independent approach performs text normalization with several passes on the text. All of the processes take the raw text as input and output the normalized/cleaned result independently. The cascaded approach also performs normalization in several passes on the text. Each process carries out cleaning/normalization from the output of the previous process.

4.5 Advantages

Our method offers some advantages.

(1) As indicated, the text normalization tasks are interdependent. The cascaded approach or the independent approach cannot simultaneously perform the tasks. In contrast, our method can effectively overcome the drawback by employing a unified framework and achieve more accurate performances.

(2) There are many specific types of errors one must correct in text normalization. As shown in Figure 1, there exist four types of errors with each type having several correction results. If one defines a specialized model or rule to handle each of the cases, the number of needed models will be extremely large and thus the text normalization processing will be impractical. In contrast, our method naturally formalizes all the tasks as assignments of different types of tags and trains a unified model to tackle all the problems at once.

5 Experimental Results

5.1 Experiment Setting

Data Sets

We used email data in our experiments. We randomly chose in total 5,000 posts (i.e., emails) from 12 newsgroups. DC, Ontology, NLP, and ML are from newsgroups at Google (http://groups-beta.google.com/groups). Jena is a newsgroup at Yahoo (http://groups.yahoo.com/group/jena-dev). Weka is a newsgroup at Waikato University (https://list.scms.waikato.ac.nz). Protégé and OWL are from a project at Stanford University (http://protege.stanford.edu/). Mobility, WinServer, Windows, and PSS are email collections from a company.

Five human annotators conducted normalization on the emails. A spec was created to guide the annotation process. All the errors in the emails were labeled and corrected. For disagreements in the annotation, we conducted "majority voting". For example, extra line breaks, extra spaces, and extra punctuation marks in the emails were labeled. Unnecessary tokens were deleted. Missing spaces and missing punctuation marks were added and marked. Mistakenly cased words, misspelled words, and misused punctuation marks were corrected. Furthermore, paragraph boundaries and sentence boundaries were also marked. The noises fell into the categories defined in Table 1.

Table 4 shows the statistics in the data sets. From the table, we can see that a large number of noises (41,407) exist in the emails. We can also see that the major noise types are extra line breaks, extra spaces, casing errors, and unnecessary tokens.

In the experiments, we conducted evaluations in terms of precision, recall, F1-measure, and accuracy (for definitions of the measures, see for example (van Rijsbergen, 1979; Lita et al., 2003)).

Implementation of Baseline Methods

We used the cascaded approach and the independent approach as baselines.

For the baseline methods, we defined several basic prediction subtasks: extra line break detection, extra space detection, extra punctuation mark detection, sentence boundary detection, unnecessary token detection, and case restoration. We compared the performances of our method with those of the baseline methods on the subtasks.

Data Set	Number of Email	Number of Noises	Extra Line Break	Extra Space	Extra Punc.	Missing Space	Missing Punc.	Casing Error	Spelling Error	Misused Punc.	Unnecessary Token	Number of Paragraph Boundary	Number of Sentence Boundary
DC	100	702	476	31	8	3	24	53	14	2	91	457	291
Ontology	100	2,731	2,132	24	3	10	68	205	79	15	195	677	1,132
NLP	60	861	623	12	1	3	23	135	13	2	49	244	296
ML	40	980	868	17	0	2	13	12	7	0	61	240	589
Jena	700	5,833	3,066	117	42	38	234	888	288	59	1,101	2,999	1,836
Weka	200	1,721	886	44	0	30	37	295	77	13	339	699	602
Protégé	700	3,306	1,770	127	48	151	136	552	116	9	397	1,645	1,035
OWL	300	1,232	680	43	24	47	41	152	44	3	198	578	424
Mobility	400	2,296	1,292	64	22	35	87	495	92	8	201	891	892
WinServer	400	3,487	2,029	59	26	57	142	822	121	21	210	1,232	1,151
Windows	1,000	9,293	3,416	3,056	60	116	348	1,309	291	67	630	3,581	2,742
PSS	1,000	8,965	3,348	2,880	59	153	296	1,331	276	66	556	3,411	2,590
Total	5,000	41,407	20,586	6,474	293	645	1,449	6,249	1,418	265	4,028	16,654	13,580

Table 4. Statistics on data sets

For the case restoration subtask (processing on token sequence), we employed the TrueCasing method (Lita et al., 2003). The method estimates a tri-gram language model using a large data corpus with correctly cased words and then makes use of the model in case restoration. We also employed Conditional Random Fields to perform case restoration, for comparison purposes. The CRF based casing method estimates a conditional probabilistic model using the same data and the same tags defined in TrueCasing.

For unnecessary token deletion, we used rules as follows. If a token consists of non-ASCII characters or consecutive duplicate characters, such as '===', then we identify it as an unnecessary token.

For each of the other subtasks, we exploited the classification approach. For example, in extra line break detection, we made use of a classification model to identify whether or not a line break is a paragraph ending. We employed Support Vector Machines (SVM) as the classification model (Vapnik, 1998). In the classification model we utilized the same features as those in our unified model (see Table 3 for details).

In the cascaded approach, the prediction tasks are performed in sequence, where the output of each task becomes the input of each immediately following task. The order of the prediction tasks is: (1) Extra line break detection: Is a line break a paragraph ending? It then separates the text into paragraphs using the remaining line breaks. (2) Extra space detection: Is a space an extra space? (3) Extra punctuation mark detection: Is a punctuation mark a noise? (4) Sentence boundary detection: Is a punctuation mark a sentence boundary? (5) Unnecessary token deletion: Is a token an unnecessary token? (6) Case restoration. Each of steps (1) to (4) uses a classification model (SVM), step (5) uses rules, whereas step (6) uses either a language model (TrueCasing) or a CRF model (CRF).

In the independent approach, we perform the prediction tasks independently. When there is a conflict between the outcomes of two classifiers, we adopt the result of the latter classifier, as determined by the order of classifiers in the cascaded approach.

To test how dependencies between different types of noises affect the performance of normalization, we also conducted experiments using the unified model by removing the transition features.

Implementation of Our Method

In the implementation of our method, we used the tool CRF++, available at http://chasen.org/~taku/software/CRF++/. We made use of all the default settings of the tool in the experiments.

5.2 Text Normalization Experiments

Results

We evaluated the performances of our method (Unified) and the baseline methods (Cascaded and Independent) on the 12 data sets. Table 5 shows the five-fold cross-validation results. Our method outperforms the two baseline methods.

Table 6 shows the overall performances of text normalization by our method and the two baseline methods. We see that our method outperforms the two baseline methods. It can also be seen that the performance of the unified method decreases when removing the transition features (Unified w/o Transition Features).

We conducted sign tests for each subtask on the results, which indicate that all the improvements of Unified over Cascaded and Independent are statistically significant ($p << 0.01$).

Detection Task		Prec.	Rec.	F1	Acc.
Extra Line Break	Independent	95.16	91.52	93.30	93.81
	Cascaded	95.16	91.52	93.30	93.81
	Unified	93.87	93.63	**93.75**	**94.53**
Extra Space	Independent	91.85	94.64	93.22	99.87
	Cascaded	94.54	94.56	94.55	99.89
	Unified	95.17	93.98	**94.57**	**99.90**
Extra Punctuation Mark	Independent	88.63	82.69	85.56	99.66
	Cascaded	87.17	85.37	86.26	99.66
	Unified	90.94	84.84	**87.78**	**99.71**
Sentence Boundary	Independent	98.46	99.62	99.04	98.36
	Cascaded	98.55	99.20	98.87	98.08
	Unified	98.76	99.61	**99.18**	**98.61**
Unnecessary Token	Independent	72.51	100.0	84.06	84.27
	Cascaded	72.51	100.0	84.06	84.27
	Unified	98.06	95.47	**96.75**	**96.18**
Case Restoration (TrueCasing)	Independent	27.32	87.44	41.63	96.22
	Cascaded	28.04	88.21	42.55	96.35
Case Restoration (CRF)	Independent	84.96	62.79	72.21	99.01
	Cascaded	85.85	63.99	73.33	99.07
	Unified	86.65	67.09	**75.63**	**99.21**

Table 5. Performances of text normalization (%)

Text Normalization	Prec.	Rec.	F1	Acc.
Independent (TrueCasing)	69.54	91.33	78.96	97.90
Independent (CRF)	85.05	92.52	88.63	98.91
Cascaded (TrueCasing)	70.29	92.07	79.72	97.88
Cascaded (CRF)	85.06	92.70	88.72	98.92
Unified w/o Transition Features	86.03	93.45	89.59	99.01
Unified	**86.46**	**93.92**	**90.04**	**99.05**

Table 6. Performances of text normalization (%)

Discussions

Our method outperforms the independent method and the cascaded method in all the subtasks, especially in the subtasks that have strong dependencies with each other, for example, sentence boundary detection, extra punctuation mark detection, and case restoration.

The cascaded method suffered from ignorance of the dependencies between the subtasks. For example, there were 3,314 cases in which sentence boundary detection needs to use the results of extra line break detection, extra punctuation mark detection, and case restoration. However, in the cascaded method, sentence boundary detection is conducted after extra punctuation mark detection and before case restoration, and thus it cannot leverage the results of case restoration. Furthermore, errors of extra punctuation mark detection can lead to errors in sentence boundary detection.

The independent method also cannot make use of dependencies across different subtasks, because it conducts all the subtasks from the raw input data. This is why for detection of extra space, extra punctuation mark, and casing error, the independent method cannot perform as well as our method.

Our method benefits from the ability of modeling dependencies between subtasks. We see from Table 6 that by leveraging the dependencies, our method can outperform the method without using dependencies (Unified w/o Transition Features) by 0.62% in terms of F1-measure.

Here we use the example in Figure 1 to show the advantage of our method compared with the independent and the cascaded methods. With normalization by the independent method, we obtain:

I'm thinking about buying a pocket PC device for my wife this Christmas, The worry that I have is that she won't be able to sync it to her outlook express contacts.//

With normalization by the cascaded method, we obtain:

I'm thinking about buying a pocket PC device for my wife this Christmas, the worry that I have is that she won't be able to sync it to her outlook express contacts.//

With normalization by our method, we obtain:

I'm thinking about buying a Pocket PC device for my wife this Christmas.// The worry that I have is that she won't be able to sync it to her Outlook Express contacts.//

The independent method can correctly deal with some of the errors. For instance, it can capitalize the first word in the first and the third line, remove extra periods in the fifth line, and remove the four extra line breaks. However, it mistakenly removes the period in the second line and it cannot restore the cases of some words, for example 'pocket' and 'outlook express'.

In the cascaded method, each process carries out cleaning/normalization from the output of the previous process and thus can make use of the cleaned/normalized results from the previous process. However, errors in the previous processes will also propagate to the later processes. For example, the cascaded method mistakenly removes the period in the second line. The error allows case restoration to make the error of keeping the word 'the' in lower case.

TrueCasing-based methods for case restoration suffer from low precision (27.32% by Independent and 28.04% by Cascaded), although their recalls are high (87.44% and 88.21% respectively). There are two reasons: 1) About 10% of the errors in Cascaded are due to errors of sentence boundary detection and extra line break detection in previous steps; 2) The two baselines tend to restore cases of words to the forms having higher probabilities in the data set and cannot take advantage of the dependencies with the other normalization subtasks. For example, 'outlook' was restored to first letter capitalized in both 'Outlook Express' and 'a pleasant outlook'. Our method can take advantage of the dependencies with other subtasks and thus correct 85.01% of the errors that the two baseline methods cannot handle. Cascaded and Independent methods employing CRF for case restoration improve the accuracies somewhat. However, they are still inferior to our method.

Although we have conducted error analysis on the results given by our method, we omit the details here due to space limitation and will report them in a future expanded version of this paper.

We also compared the speed of our method with those of the independent and cascaded methods. We tested the three methods on a computer with two 2.8G Dual-Core CPUs and three Gigabyte memory. On average, it needs about 5 hours for training the normalization models using our method and 25 seconds for tagging in the cross-validation experiments. The independent and the cascaded methods (with TrueCasing) require less time for training (about 2 minutes and 3 minutes respectively) and for tagging (several seconds). This indicates that the efficiency of our method still needs improvement.

6 Conclusion

In this paper, we have investigated the problem of text normalization, an important issue for natural language processing. We have first defined the problem as a task consisting of noise elimination and boundary detection subtasks. We have then proposed a unified tagging approach to perform the task, specifically to treat text normalization as assigning tags representing deletion, preservation, or replacement of the tokens in the text. Experiments show that our approach significantly outperforms the two baseline methods for text normalization.

References

E. Brill and R. C. Moore. 2000. An Improved Error Model for Noisy Channel Spelling Correction, *Proc. of ACL 2000.*

V. R. Carvalho and W. W. Cohen. 2004. Learning to Extract Signature and Reply Lines from Email, *Proc. of CEAS 2004.*

K. Church and W. Gale. 1991. Probability Scoring for Spelling Correction, *Statistics and Computing*, Vol. 1.

A. Clark. 2003. Pre-processing Very Noisy Text, *Proc. of Workshop on Shallow Processing of Large Corpora*.

A. R. Golding and D. Roth. 1996. Applying Winnow to Context-Sensitive Spelling Correction, *Proc. of ICML'1996*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data, *Proc. of ICML 2001*.

L. V. Lita, A. Ittycheriah, S. Roukos, and N. Kambhatla. 2003. tRuEcasIng, *Proc. of ACL 2003*.

E. Mays, F. J. Damerau, and R. L. Mercer. 1991. Context Based Spelling Correction, *Information Processing and Management*, Vol. 27, 1991.

A. Mikheev. 2000. Document Centered Approach to Text Normalization, *Proc. SIGIR 2000.*

A. Mikheev. 2002. Periods, Capitalized Words, etc. *Computational Linguistics*, Vol. 28, 2002.

E. Minkov, R. C. Wang, and W. W. Cohen. 2005. Extracting Personal Names from Email: Applying Named Entity Recognition to Informal Text, *Proc. of EMNLP/HLT-2005*.

D. D. Palmer and M. A. Hearst. 1997. Adaptive Multilingual Sentence Boundary Disambiguation, *Computational Linguistics*, Vol. 23.

C.J. van Rijsbergen. 1979. *Information Retrieval*. Butterworths, London.

R. Sproat, A. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards. 1999. Normalization of non-standard words, *WS'99 Final Report.* http://www.clsp.jhu.edu/ws99/projects/normal/.

J. Tang, H. Li, Y. Cao, and Z. Tang. 2005. Email data cleaning, *Proc. of SIGKDD'2005.*

V. Vapnik. 1998. *Statistical Learning Theory*, Springer.

W. Wong, W. Liu, and M. Bennamoun. 2007. Enhanced Integrated Scoring for Cleaning Dirty Texts, *Proc. of IJCAI-2007 Workshop on Analytics for Noisy Unstructured Text Data.*

Sparse Information Extraction: Unsupervised Language Models to the Rescue

Doug Downey, Stefan Schoenmackers, and Oren Etzioni
Turing Center, Department of Computer Science and Engineering
University of Washington, Box 352350
Seattle, WA 98195, USA
{ddowney,stef,etzioni}@cs.washington.edu

Abstract

Even in a massive corpus such as the Web, a substantial fraction of extractions appear infrequently. This paper shows how to assess the correctness of *sparse* extractions by utilizing unsupervised language models. The REALM system, which combines HMM-based and n-gram-based language models, ranks candidate extractions by the likelihood that they are correct. Our experiments show that REALM reduces extraction error by 39%, on average, when compared with previous work.

Because REALM pre-computes language models based on its corpus and does not require *any* hand-tagged seeds, it is far more scalable than approaches that learn models for each individual relation from hand-tagged data. Thus, REALM is ideally suited for open information extraction where the relations of interest are not specified in advance and their number is potentially vast.

1 Introduction

Information Extraction (IE) from text is far from infallible. In response, researchers have begun to exploit the redundancy in massive corpora such as the Web in order to assess the veracity of extractions (*e.g.*, (Downey et al., 2005; Etzioni et al., 2005; Feldman et al., 2006)). In essence, such methods utilize *extraction patterns* to generate candidate extractions (*e.g.*, "Istanbul") and then assess each candidate by computing co-occurrence statistics between the extraction and words or phrases indicative of class membership (*e.g.*, "cities such as").

However, Zipf's Law governs the distribution of extractions. Thus, even the Web has limited redundancy for less prominent instances of relations. Indeed, 50% of the extractions in the data sets employed by (Downey et al., 2005) appeared only once. As a result, Downey *et al.*'s model, and related methods, had no way of assessing which extraction is more likely to be correct for fully half of the extractions. This problem is particularly acute when moving beyond unary relations. We refer to this challenge as the task of *assessing sparse extractions*.

This paper introduces the idea that *language modeling* techniques such as n-gram statistics (Manning and Schütze, 1999) and HMMs (Rabiner, 1989) can be used to effectively assess sparse extractions. The paper introduces the REALM system, and highlights its unique properties. Notably, REALM does not require *any* hand-tagged seeds, which enables it to scale to *Open IE*—extraction where the relations of interest are not specified in advance, and their number is potentially vast (Banko et al., 2007).

REALM is based on two key hypotheses. The *KnowItAll hypothesis* is that extractions that occur more frequently in distinct sentences in the corpus are more likely to be correct. For example, the hypothesis suggests that the argument pair (`Giuliani`, `New York`) is relatively likely to be appropriate for the `Mayor` relation, simply because this pair is extracted for the `Mayor` relation relatively frequently. Second, we employ an instance of the *distributional hypothesis* (Harris, 1985), which

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 696–703,
Prague, Czech Republic, June 2007.

can be phrased as follows: different instances of the same semantic relation tend to appear in similar textual contexts. We assess sparse extractions by comparing the contexts in which they appear to those of more common extractions. Sparse extractions whose contexts are more similar to those of common extractions are judged more likely to be correct based on the conjunction of the KnowItAll and the distributional hypotheses.

The contributions of the paper are as follows:

- The paper introduces the insight that the subfield of language modeling provides unsupervised methods that can be leveraged to assess sparse extractions. These methods are more scalable than previous assessment techniques, and require no hand tagging whatsoever.
- The paper introduces an HMM-based technique for checking whether two arguments are of the proper type for a relation.
- The paper introduces a *relational* n-gram model for the purpose of determining whether a sentence that mentions multiple arguments actually expresses a particular relationship between them.
- The paper introduces a novel language-modeling system called REALM that combines both HMM-based models and relational n-gram models, and shows that REALM reduces error by an average of 39% over previous methods, when applied to sparse extraction data.

The remainder of the paper is organized as follows. Section 2 introduces the IE assessment task, and describes the REALM system in detail. Section 3 reports on our experimental results followed by a discussion of related work in Section 4. Finally, we conclude with a discussion of scalability and with directions for future work.

2 IE Assessment

This section formalizes the *IE assessment* task and describes the REALM system for solving it. An IE assessor takes as input a list of candidate extractions meant to denote instances of a relation, and outputs a *ranking* of the extractions with the goal that correct extractions rank higher than incorrect ones. A *correct* extraction is defined to be a true instance of the relation mentioned in the input text.

More formally, the list of candidate extractions for a relation R is denoted as $E_R = \{(a_1, b_1), \ldots, (a_m, b_m)\}$. An extraction ($a_i$, b_i) is an ordered pair of strings. The extraction is *correct* if and only if the relation R holds between the arguments named by a_i and b_i. For example, for $R =$ `Headquartered`, a pair (a_i, b_i) is correct *iff* there exists an organization a_i that is in fact headquartered in the location b_i.[1]

E_R is generated by applying an extraction mechanism, typically a set of extraction "patterns", to each sentence in a corpus, and recording the results. Thus, many elements of E_R are identical extractions derived from different sentences in the corpus.

This task definition is notable for the minimal inputs required—IE assessment does not require knowing the relation name nor does it require hand-tagged seed examples of the relation. Thus, an IE Assessor is applicable to Open IE.

2.1 System Overview

In this section, we describe the REALM system, which utilizes language modeling techniques to perform IE Assessment.

REALM takes as input a set of extractions E_R, and outputs a ranking of those extractions. The algorithm REALM follows is outlined in Figure 1. REALM begins by automatically selecting from E_R a set of *bootstrapped seeds* S_R intended to serve as correct examples of the relation R. REALM utilizes the KnowItAll hypothesis, setting S_R equal to the h elements in E_R extracted most frequently from the underlying corpus. This results in a noisy set of seeds, but the methods that use these seeds are noise tolerant.

REALM then proceeds to rank the remaining (non-seed) extractions by utilizing two language-modeling components. An *n-gram language model* is a probability distribution $P(w_1, ..., w_n)$ over consecutive word sequences of length n in a corpus. Formally, if we assume a seed (s_1, s_2) is a correct extraction of a relation R, the distributional hypothesis states that the *context distribution* around the seed extraction, $P(w_1, ..., w_n | w_i = s_1, w_j = s_2)$ for $1 \leq i, j \leq n$ tends to be "more similar" to

[1] For clarity, our discussion focuses on relations between pairs of arguments. However, the methods we propose can be extended to relations of any arity.

$P(w_1, ..., w_n | w_i = e_1, w_j = e_2)$ when the extraction (e_1, e_2) is correct. Naively comparing context distributions is problematic, however, because the arguments to a relation often appear separated by several intervening words. In our experiments, we found that when relation arguments appear together in a sentence, 75% of the time the arguments are separated by at least three words. This implies that n must be large, and for sparse argument pairs it is not possible to estimate such a large language model accurately, because the number of modeling parameters is proportional to the vocabulary size raised to the nth power. To mitigate sparsity, REALM utilizes smaller language models in its two components as a means of "backing-off' from estimating context distributions explicitly, as described below.

First, REALM utilizes an HMM to estimate whether each extraction has arguments of the proper type for the relation. Each relation R has a set of types for its arguments. For example, the relation `AuthorOf(a, b)` requires that its first argument be an author, and that its second be some kind of written work. Knowing whether extracted arguments are of the proper type for a relation can be quite informative for assessing extractions. The challenge is, however, that this type information is *not* given to the system since the relations (and the types of the arguments) are not known in advance. REALM solves this problem by comparing the distributions of the seed arguments and extraction arguments. Type checking mitigates data sparsity by leveraging *every* occurrence of the individual extraction arguments in the corpus, rather than only those cases in which argument pairs occur near each other.

Although argument type checking is invaluable for extraction assessment, it is not sufficient for extracting relationships between arguments. For example, an IE system using only type information might determine that `Intel` is a corporation and that `Seattle` is a city, and therefore erroneously conclude that `Headquartered(Intel, Seattle)` is correct. Thus, REALM's second step is to employ an n-gram-based language model to assess whether the extracted arguments share the appropriate relation. Again, this information is not given to the system, so REALM compares the context distributions of the extractions to those of the seeds. As described in Section 2.3, REALM employs a relational n-gram language model in order to accurately compare context distributions when extractions are sparse.

REALM(*Extractions* $E_R = \{e_1, ..., e_m\}$)
  S_R = the h most frequent extractions in E_R
  U_R = E_R - S_R
  $TypeRankings(U_R) \leftarrow$ HMM-T(S_R, U_R)
  $RelationRankings(U_R) \leftarrow$ REL-GRAMS(S_R, U_R)
  return a ranking of E_R with the elements of S_R at the top (ranked by frequency) followed by the elements of $U_R = \{u_1, ..., u_{m-h}\}$ ranked in ascending order of $TypeRanking(u_i) * RelationRanking(u_i)$.

Figure 1: Pseudocode for REALM at run-time. The language models used by the HMM-T and REL-GRAMS components are constructed in a preprocessing step.

REALM executes the type checking and relation assessment components separately; each component takes the seed and non-seed extractions as arguments and returns a ranking of the non-seeds. REALM then combines the two components' assessments into a single ranking. Although several such combinations are possible, REALM simply ranks the extractions in ascending order of the product of the ranks assigned by the two components. The following subsections describe REALM's two components in detail.

We identify the proper nouns in our corpus using the LEX method (Downey et al., 2007). In addition to locating the proper nouns in the corpus, LEX also concatenates each multi-token proper noun (*e.g.*,`Los Angeles`) together into a single token. Both of REALM's components construct language models from this tokenized corpus.

2.2 Type Checking with HMM-T

In this section, we describe our type-checking component, which takes the form of a Hidden Markov Model and is referred to as HMM-T. HMM-T ranks the set U_R of non-seed extractions, with a goal of ranking those extractions with arguments of proper type for R above extractions containing type errors. Formally, let U_{Ri} denote the set of the ith arguments of the extractions in U_R. Let S_{Ri} be defined similarly for the seed set S_R.

Our type checking technique exploits the distributional hypothesis—in this case, the intuition that

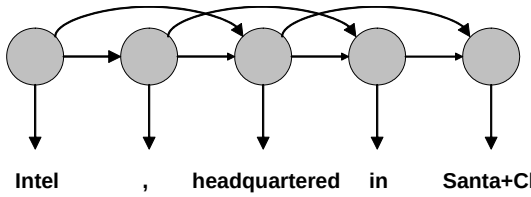


Figure 2: Graphical model employed by HMM-T. Shown is the case in which k = 2. Corpus pre-processing results in the proper noun `Santa Clara` being concatenated into a single token.

extraction arguments in U_{Ri} of the proper type will likely appear in contexts similar to those in which the seed arguments S_{Ri} appear. In order to identify terms that are distributionally similar, we train a probabilistic generative Hidden Markov Model (HMM), which treats each token in the corpus as generated by a single hidden state variable. Here, the hidden states take integral values from $\{1, \ldots, T\}$, and each hidden state variable is itself generated by some number k of previous hidden states.[2] Formally, the joint distribution of the corpus, represented as a vector of tokens $\mathbf{w}$, given a corresponding vector of states $\mathbf{t}$ is:

$$P(\mathbf{w}|\mathbf{t}) = \prod_i P(w_i|t_i)P(t_i|t_{i-1}, \ldots, t_{i-k}) \quad (1)$$

The distributions on the right side of Equation 1 can be learned from a corpus in an unsupervised manner, such that words which are distributed similarly in the corpus tend to be generated by similar hidden states (Rabiner, 1989). The generative model is depicted as a Bayesian network in Figure 2. The figure also illustrates the one way in which our implementation is distinct from a standard HMM, namely that proper nouns are detected *a priori* and modeled as single tokens (*e.g.*, `Santa Clara` is generated by a single hidden state). This allows the type checker to compare the state distributions of different proper nouns directly, even when the proper nouns contain differing numbers of words.

To generate a ranking of U_R using the learned HMM parameters, we rank the arguments e_i according to how similar their *state distributions* $P(t|e_i)$ are to those of the seed arguments.[3] Specifically, we define a function:

$$f(e) = \sum_{e_i \in e} KL(\frac{\sum_{w' \in S_{Ri}} P(t|w')}{|S_{Ri}|}, P(t|e_i)) \quad (2)$$

where KL represents KL divergence, and the outer sum is taken over the arguments e_i of the extraction e. We rank the elements of U_R in ascending order of $f(e)$.

HMM-T has two advantages over a more traditional type checking approach of simply counting the number of times in the corpus that each extraction appears in a context in which a seed also appears (*cf.* (Ravichandran et al., 2005)). The first advantage of HMM-T is efficiency, as the traditional approach involves a computationally expensive step of retrieving the potentially large set of contexts in which the extractions and seeds appear. In our experiments, using HMM-T instead of a context-based approach results in a 10-50x reduction in the amount of data that is retrieved to perform type checking. Secondly, on sparse data HMM-T has the potential to improve type checking accuracy. For example, consider comparing `Pickerington`, a sparse candidate argument of the type `City`, to the seed argument `Chicago`, for which the following two phrases appear in the corpus:

(i) "Pickerington, Ohio"
(ii) "Chicago, Illinois"

In these phrases, the textual contexts surrounding `Chicago` and `Pickerington` are not identical, so to the traditional approach these contexts offer no evidence that `Pickerington` and `Chicago` are of the same type. For a sparse token like `Pickerington`, this is problematic because the token may *never* occur in a context that precisely matches that of a seed. In contrast, in the HMM, the non-sparse tokens `Ohio` and `Illinois` are likely to have similar state distributions, as they are both the names of U.S. States. Thus, in the state space employed by the HMM, the contexts in phrases (i) and (ii) are in fact quite similar, allowing HMM-T to detect that `Pickerington` and `Chicago` are likely of the same type. Our experiments quantify the performance improvements that HMM-T of-

[2] Our implementation makes the simplifying assumption that each sentence in the corpus is generated independently.

[3] The distribution $P(t|e_i)$ for any e_i can be obtained from the HMM parameters using Bayes Rule.

fers over the traditional approach for type checking sparse data.

The time required to learn HMM-T's parameters scales proportional to T^{k+1} times the corpus size. Thus, for tractability, HMM-T uses a relatively small state space of $T = 20$ states and a limited k value of 3. While these settings are sufficient for type checking (*e.g.*, determining that `Santa Clara` is a city) they are too coarse-grained to assess relations between arguments (*e.g.*, determining that `Santa Clara` is the particular city in which `Intel` is headquartered). We now turn to the REL-GRAMS component, which performs the latter task.

2.3 Relation Assessment with REL-GRAMS

REALM's relation assessment component, called REL-GRAMS, tests whether the extracted arguments have a desired relationship, but given REALM's minimal input it has no *a priori* information about the relationship. REL-GRAMS relies instead on the distributional hypothesis to test each extraction.

As argued in Section 2.1, it is intractable to build an accurate language model for context distributions surrounding sparse argument pairs. To overcome this problem, we introduce *relational* n-gram models. Rather than simply modeling the context distribution around a given argument, a relational n-gram model specifies separate context distributions for an arguments *conditioned on* each of the other arguments with which it appears. The relational n-gram model allows us to estimate context distributions for pairs of arguments, even when the arguments do not appear together within a fixed window of n words. Further, by considering only consecutive argument pairs, the number of distinct argument pairs in the model grows at most linearly with the number of sentences in the corpus. Thus, the relational n-gram model can scale.

Formally, for a pair of arguments (e_1, e_2), a relational n-gram model estimates the distributions $P(w_1, ..., w_n | w_i = e_1, e_1 \leftrightarrow e_2)$ for each $1 \leq i \leq n$, where the notation $e_1 \leftrightarrow e_2$ indicates the event that e_2 is the next argument to either the right or the left of e_1 in the corpus.

REL-GRAMS begins by building a relational n-gram model of the arguments in the corpus. For notational convenience, we represent the model's distributions in terms of "context vectors" for each pair of arguments. Formally, for a given sentence containing arguments e_1 and e_2 consecutively, we define a *context* of the ordered pair (e_1, e_2) to be any window of n tokens around e_1. Let $C = \{c_1, c_2, ..., c_{|C|}\}$ be the set of all contexts of all argument pairs found in the corpus.[4] For a pair of arguments (e_j, e_k), we model their relationship using a $|C|$ dimensional context vector $v_{(e_j,e_k)}$, whose i-th dimension corresponds to the number of times context c_i occurred with the pair (e_j, e_k) in the corpus. These context vectors are similar to document vectors from Information Retrieval (IR), and we leverage IR research to compare them, as described below.

To assess each extraction, we determine how similar its context vector is to a canonical seed vector (created by summing the context vectors of the seeds). While there are many potential methods for determining similarity, in this work we rank extractions by decreasing values of the BM25 distance metric. BM25 is a TF-IDF variant introduced in TREC-3(Robertson et al., 1992), which outperformed both the standard cosine distance and a smoothed KL divergence on our data.

3 Experimental Results

This section describes our experiments on IE assessment for sparse data. We start by describing our experimental methodology, and then present our results. The first experiment tests the hypothesis that HMM-T outperforms an n-gram-based method on the task of type checking. The second experiment tests the hypothesis that REALM outperforms multiple approaches from previous work, and also outperforms each of its HMM-T and REL-GRAMS components taken in isolation.

3.1 Experimental Methodology

The corpus used for our experiments consisted of a sample of sentences taken from Web pages. From an initial crawl of nine million Web pages, we selected sentences containing relations between proper nouns. The resulting text corpus consisted of about

[4] Pre-computing the set C requires identifying in advance the potential relation arguments in the corpus. We consider the proper nouns identified by the LEX method (see Section 2.1) to be the potential arguments.

three million sentences, and was tokenized as described in Section 2. For tractability, before and after performing tokenization, we replaced each token occurring fewer than five times in the corpus with one of two "unknown word" markers (one for capitalized words, and one for uncapitalized words). This preprocessing resulted in a corpus containing about sixty-five million total tokens, and 214,787 unique tokens.

We evaluated performance on four relations: `Conquered`, `Founded`, `Headquartered`, and `Merged`. These four relations were chosen because they typically take proper nouns as arguments, and included a large number of sparse extractions. For each relation R, the candidate extraction list E_R was obtained using TEXTRUNNER (Banko et al., 2007). TEXTRUNNER is an IE system that computes an index of all extracted relationships it recognizes, in the form of (object, predicate, object) triples. For each of our target relations, we executed a single query to the TEXTRUNNER index for extractions whose predicate contained a phrase indicative of the relation (*e.g.*, "founded by", "headquartered in"), and the results formed our extraction list. For each relation, the 10 most frequent extractions served as bootstrapped seeds. All of the non-seed extractions were sparse (no argument pairs were extracted more than twice for a given relation). These test sets contained a total of 361 extractions.

3.2 Type Checking Experiments

As discussed in Section 2.2, on sparse data HMM-T has the potential to outperform type checking methods that rely on textual similarities of context vectors. To evaluate this claim, we tested the HMM-T system against an N-GRAMS type checking method on the task of type-checking the arguments to a relation. The N-GRAMS method compares the context vectors of extractions in the same way as the REL-GRAMS method described in Section 2.3, but is not relational (N-GRAMS considers the distribution of each extraction argument independently, similar to HMM-T). We tagged an extraction as type correct *iff* both arguments were valid for the relation, ignoring whether the relation held between the arguments.

The results of our type checking experiments are shown in Table 1. For all types, HMM-T outperforms N-GRAMS, and HMM-T reduces error (measured in missing area under the precision/recall curve) by 46%. The performance difference on each relation is statistically significant ($p < 0.01$, two-sampled t-test), using the methodology for measuring the standard deviation of area under the precision/recall curve given in (Richardson and Domingos, 2006). N-GRAMS, like REL-GRAMS, employs the BM-25 metric to measure distributional similarity between extractions and seeds. Replacing BM-25 with cosine distance cuts HMM-T's advantage over N-GRAMS, but HMM-T's error rate is still 23% lower on average.

Type	HMM-T	N-GRAMS
`Conquered`	**0.917**	0.767
`Founded`	**0.827**	0.636
`Headquartered`	**0.734**	0.589
`Merged`	**0.920**	0.854
Average	**0.849**	0.712

Table 1: Type Checking Performance. Listed is area under the precision/recall curve. HMM-T outperforms N-GRAMS for all relations, and reduces the error in terms of missing area under the curve by 46% on average.

3.3 Experiments with REALM

The REALM system combines the type checking and relation assessment components to assess extractions. Here, we test the ability of REALM to improve the ranking of a state of the art IE system, TEXTRUNNER. For these experiments, we evaluate REALM against the TEXTRUNNER frequency-based ordering, a pattern-learning approach, and the HMM-T and REL-GRAMS components taken in isolation. The TEXTRUNNER frequency-based ordering ranks extractions in decreasing order of their extraction frequency, and importantly, for our task this ordering is essentially equivalent to that produced by the "Urns" (Downey et al., 2005) and Pointwise Mutual Information (Etzioni et al., 2005) approaches employed in previous work.

The pattern-learning approach, denoted as PL, is modeled after Snowball (Agichtein, 2006). The algorithm and parameter settings for PL were those manually tuned for the `Headquartered` relation in previous work (Agichtein, 2005). A sensitivity analysis of these parameters indicated that the re-

	Conquered	Founded	Headquartered	Merged	Average
Avg. Prec.	0.698	0.578	0.400	0.742	0.605
TEXTRUNNER	0.738	0.699	0.710	0.784	0.733
PL	0.885	0.633	0.651	0.852	0.785
PL+ HMM-T	0.883	0.722	0.727	0.900	0.808
HMM-T	0.830	0.776	0.678	0.864	0.787
REL-GRAMS	**0.929 (39%)**	0.713	0.758	0.886	0.822
REALM	0.907 (19%)	**0.781 (27%)**	**0.810 (35%)**	**0.908 (38%)**	**0.851 (39%)**

Table 2: Performance of REALM for assessment of sparse extractions. Listed is area under the precision/recall curve for each method. In parentheses is the percentage reduction in error over the strongest baseline method (TEXTRUNNER or PL) for each relation. "Avg. Prec." denotes the fraction of correct examples in the test set for each relation. REALM outperforms its REL-GRAMS and HMM-T components taken in isolation, as well as the TEXTRUNNER and PL systems from previous work.

sults are sensitive to the parameter settings. However, we found no parameter settings that performed significantly better, and many settings performed significantly worse. As such, we believe our results reasonably reflect the performance of a pattern learning system on this task. Because PL performs relation assessment, we also attempted combining PL with HMM-T in a hybrid method (PL+ HMM-T) analogous to REALM.

The results of these experiments are shown in Table 2. REALM outperforms the TEXTRUNNER and PL baselines for all relations, and reduces the missing area under the curve by an average of 39% relative to the strongest baseline. The performance differences between REALM and TEXTRUNNER are statistically significant for all relations, as are differences between REALM and PL for all relations except Conquered ($p < 0.01$, two-sampled t-test). The hybrid REALM system also outperforms each of its components in isolation.

4 Related Work

To our knowledge, REALM is the first system to use language modeling techniques for IE Assessment.

Redundancy-based approaches to pattern-based IE assessment (Downey et al., 2005; Etzioni et al., 2005) require that extractions appear relatively frequently with a limited set of patterns. In contrast, REALM utilizes *all* contexts to build a model of extractions, rather than a limited set of patterns. Our experiments demonstrate that REALM outperforms these approaches on sparse data.

Type checking using *named-entity taggers* has been previously shown to improve the precision of pattern-based IE systems (Agichtein, 2005; Feldman et al., 2006), but the HMM-T type-checking component we develop differs from this work in important ways. Named-entity taggers are limited in that they typically recognize only small set of types (*e.g.*, ORGANIZATION, LOCATION, PERSON), and they require hand-tagged training data for each type. HMM-T, by contrast, performs type checking for *any* type. Finally, HMM-T does not require hand-tagged training data.

Pattern learning is a common technique for extracting and assessing sparse data (*e.g.* (Agichtein, 2005; Riloff and Jones, 1999; Paşca et al., 2006)). Our experiments demonstrate that REALM outperforms a pattern learning system closely modeled after (Agichtein, 2005). REALM is inspired by pattern learning techniques (in particular, both use the distributional hypothesis to assess sparse data) but is distinct in important ways. Pattern learning techniques require substantial processing of the corpus after the relations they assess have been specified. Because of this, pattern learning systems are unsuited to Open IE. Unlike these techniques, REALM pre-computes language models which allow it to assess extractions for arbitrary relations at run-time. In essence, pattern-learning methods run in time linear in the number of relations whereas REALM's run time is constant in the number of relations. Thus, REALM scales readily to large numbers of relations whereas pattern-learning methods do not.

A second distinction of REALM is that its type checker, unlike the named entity taggers employed in pattern learning systems (*e.g.*, Snowball), can be used to identify arbitrary types. A final distinction is that the language models REALM employs require fewer parameters and heuristics than pattern learning techniques.

Similar distinctions exist between REALM and a recent system designed to assess sparse extractions by bootstrapping a *classifier* for each target relation (Feldman et al., 2006). As in pattern learning, constructing the classifiers requires substantial processing after the target relations have been specified, and a set of hand-tagged examples per relation, making it unsuitable for Open IE.

5 Conclusions

This paper demonstrated that unsupervised language models, as embodied in the REALM system, are an effective means of assessing sparse extractions.

Another attractive feature of REALM is its scalability. Scalability is a particularly important concern for *Open Information Extraction*, the task of extracting large numbers of relations that are not specified in advance. Because HMM-T and REL-GRAMS both pre-compute language models, REALM can be queried efficiently to perform IE Assessment. Further, the language models are constructed independently of the target relations, allowing REALM to perform IE Assessment even when relations are not specified in advance.

In future work, we plan to develop a probabilistic model of the information computed by REALM. We also plan to evaluate the use of non-local context for IE Assessment by integrating document-level modeling techniques (*e.g.*, Latent Dirichlet Allocation).

Acknowledgements

This research was supported in part by NSF grants IIS-0535284 and IIS-0312988, DARPA contract NBCHD030010, ONR grant N00014-05-1-0185 as well as a gift from Google. The first author is supported by an MSR graduate fellowship sponsored by Microsoft Live Labs. We thank Michele Banko, Jeff Bilmes, Katrin Kirchhoff, and Alex Yates for helpful comments.

References

E. Agichtein. 2005. *Extracting Relations From Large Text Collections*. Ph.D. thesis, Department of Computer Science, Columbia University.

E. Agichtein. 2006. Confidence estimation methods for partially supervised relation extraction. In *SDM 2006*.

M. Banko, M. Cararella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction from the web. In *Procs. of IJCAI 2007*.

D. Downey, O. Etzioni, and S. Soderland. 2005. A Probabilistic Model of Redundancy in Information Extraction. In *Procs. of IJCAI 2005*.

D. Downey, M. Broadhead, and O. Etzioni. 2007. Locating complex named entities in web text. In *Procs. of IJCAI 2007*.

O. Etzioni, M. Cafarella, D. Downey, S. Kok, 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.

R. Feldman, B. Rosenfeld, S. Soderland, and O. Etzioni. 2006. Self-supervised relation extraction from the web. In *ISMIS*, pages 755–764.

Z. Harris. 1985. Distributional structure. In J. J. Katz, editor, *The Philosophy of Linguistics*, pages 26–47. New York: Oxford University Press.

C. D. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*.

M. Paşca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. 2006. Names and similarities on the web: Fact extraction in the fast lane. In *Procs. of ACL/COLING 2006*.

L. R. Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

D. Ravichandran, P. Pantel, and E. H. Hovy. 2005. Randomized Algorithms and NLP: Using Locality Sensitive Hash Functions for High Speed Noun Clustering. In *Procs. of ACL 2005*.

M. Richardson and P. Domingos. 2006. Markov Logic Networks. *Machine Learning*, 62(1-2):107–136.

E. Riloff and R. Jones. 1999. Learning Dictionaries for Information Extraction by Multi-level Boot-strapping. In *Procs. of AAAI-99*, pages 1044–1049.

S. E. Robertson, S. Walker, M. Hancock-Beaulieu, A. Gull, and M. Lau. 1992. Okapi at TREC-3. In *Text REtrieval Conference*, pages 21–30.

Forest-to-String Statistical Translation Rules

Yang Liu , Yun Huang , Qun Liu and **Shouxun Lin**
Key Laboratory of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
P.O. Box 2704, Beijing 100080, China
{yliu,huangyun,liuqun,sxlin}@ict.ac.cn

Abstract

In this paper, we propose forest-to-string rules to enhance the expressive power of tree-to-string translation models. A forest-to-string rule is capable of capturing non-syntactic phrase pairs by describing the correspondence between multiple parse trees and one string. To integrate these rules into tree-to-string translation models, auxiliary rules are introduced to provide a generalization level. Experimental results show that, on the NIST 2005 Chinese-English test set, the tree-to-string model augmented with forest-to-string rules achieves a relative improvement of 4.3% in terms of BLEU score over the original model which allows tree-to-string rules only.

1 Introduction

The past two years have witnessed the rapid development of linguistically syntax-based translation models (Quirk et al., 2005; Galley et al., 2006; Marcu et al., 2006; Liu et al., 2006), which induce tree-to-string translation rules from parallel texts with linguistic annotations. They demonstrated very promising results when compared with the state of the art phrase-based system (Och and Ney, 2004) in the NIST 2006 machine translation evaluation [1]. While Galley et al. (2006) and Marcu et al. (2006) put emphasis on target language analysis, Quirk et al. (2005) and Liu et al. (2006) show benefits from modeling the syntax of source language.

One major problem with linguistically syntax-based models, however, is that tree-to-string rules fail to syntactify non-syntactic phrase pairs because they require a syntax tree fragment over the phrase to be syntactified. Here, we distinguish between *syntactic* and *non-syntactic* phrase pairs. By "syntactic" we mean that the phrase pair is subsumed by some syntax tree fragment. The phrase pairs without trees over them are non-syntactic. Marcu et al. (2006) report that approximately 28% of bilingual phrases are non-syntactic on their English-Chinese corpus.

We believe that it is important to make available to syntax-based models all the bilingual phrases that are typically available to phrase-based models. On one hand, phrases have been proven to be a simple and powerful mechanism for machine translation. They excel at capturing translations of short idioms, providing local re-ordering decisions, and incorporating context information straightforwardly. Chiang (2005) shows significant improvement by keeping the strengths of phrases while incorporating syntax into statistical translation. On the other hand, the performance of linguistically syntax-based models can be hindered by making use of only syntactic phrase pairs. Studies reveal that linguistically syntax-based models are sensitive to syntactic analysis (Quirk and Corston-Oliver, 2006), which is still not reliable enough to handle real-world texts due to limited size and domain of training data.

Various solutions are proposed to tackle the problem. Galley et al. (2004) handle non-constituent phrasal translation by traversing the tree upwards until reaches a node that subsumes the phrase. Marcu et al. (2006) argue that this choice is inap-

[1] See http://www.nist.gov/speech/tests/mt/

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 704–711,
Prague, Czech Republic, June 2007.

propriate because large applicability contexts are required.

For a non-syntactic phrase pair, Marcu et al. (2006) create a xRS rule headed by a pseudo, non-syntactic nonterminal symbol that subsumes the phrase and corresponding multi-headed syntactic structure; and one sibling xRS rule that explains how the non-syntactic nonterminal symbol can be combined with other genuine nonterminals so as to obtain genuine parse trees. The name of the pseudo nonterminal is designed to reflect how the corresponding rule can be fully realized. However, they neglect alignment consistency when creating sibling rules. In addition, it is hard for the naming mechanism to deal with more complex phenomena.

Liu et al. (2006) treat bilingual phrases as lexicalized TATs (Tree-to-string Alignment Template). A bilingual phrase can be used in decoding if the source phrase is subsumed by the input parse tree. Although this solution does help, only syntactic bilingual phrases are available to the TAT-based model. Moreover, it is problematic to combine the translation probabilities of bilingual phrases and TATs, which are estimated independently.

In this paper, we propose forest-to-string rules which describe the correspondence between multiple parse trees and a string. They can not only capture non-syntactic phrase pairs but also have the capability of generalization. To integrate these rules into tree-to-string translation models, auxiliary rules are introduced to provide a generalization level. As there is no pseudo node or naming mechanism, the integration of forest-to-string rules is flexible, relying only on their root nodes. The forest-to-string and auxiliary rules enable tree-to-string models to derive in a more general way, while the strengths of conventional tree-to-string rules still remain.

2 Forest-to-String Translation Rules

We define a tree-to-string rule r as a triple $\langle \tilde{T}, \tilde{S}, \tilde{A} \rangle$, which describes the alignment $\tilde{A}$ between a source parse tree $\tilde{T} = T(f_1^{J'})$ and a target string $\tilde{S} = e_1^{I'}$. A source string $f_1^{J'}$, which is the sequence of leaf nodes of $T(f_1^{J'})$, consists of both terminals (source words) and nonterminals (phrasal categories). A target string $e_1^{I'}$ is also composed of both terminals (target words) and nonterminals (placeholders). An alignment $\tilde{A}$ is defined as a subset of the Cartesian product of source and target symbol positions:

$$\tilde{A} \subseteq \{(j, i) : j = 1, \ldots, J'; i = 1, \ldots, I'\}$$

A derivation $\theta = r_1 \circ r_2 \circ \ldots \circ r_n$ is a left-most composition of translation rules that explains how a source parse tree $T = T(f_1^J)$, a target sentence $S = e_1^I$, and the word alignment A are synchronously generated. For example, Table 1 demonstrates a derivation composed of only tree-to-string rules for the $\langle T, S, A \rangle$ tuple in Figure 1 [2].

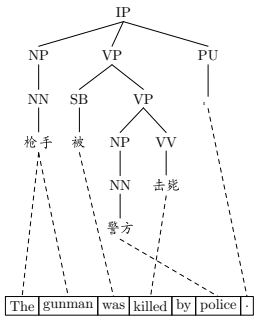


Figure 1: An English sentence aligned with a Chinese parse tree.

As we mentioned before, tree-to-string rules can not syntactify phrase pairs that are not subsumed by any syntax tree fragments. For example, for the phrase pair $\langle$"枪手 被", "*The gunman was*"$\rangle$ in Figure 1, it is impossible to extract an equivalent tree-to-string rule that subsumes the same phrase pair because valid tree-to-string rules can not be multi-headed.

To address this problem, we propose *forest-to-string rules*[3] to subsume the non-syntactic phrase pairs. A forest-to-string rule r [4] is a triple $\langle \tilde{F}, \tilde{S}, \tilde{A} \rangle$, which describes the alignment $\tilde{A}$ between K source parse trees $\tilde{F} = \tilde{T}_1^K$ and a target string $\tilde{S}$. The source string $f_1^{J'}$ is therefore the sequence of leaf nodes of $\tilde{F}$.

Auxiliary rules are introduced to integrate forest-to-string rules into tree-to-string translation models. An auxiliary rule is a special unlexicalized tree-to-string rule that allows multiple source nonterminals

[2] We use "X" to denote a nonterminal in the target string. If there are more than one nonterminals, they are indexed.

[3] The term "forest" refers to an ordered and finite set of trees.

[4] We still use "r" to represent a forest-to-string rule to reduce notational overhead.

No.	Rule		
(1)	(IP (NP) (VP) (PU))	$X_1\ X_2\ X_3$	1:1 2:2 3:3
(2)	(NP (NN 枪手))	The gunman	1:1 1:2
(3)	(VP (SB 被) (VP (NP (NN)) (VV 击毙)))	was killed by X	1:1 2:4 3:2
(4)	(NN 警方)	police	1:1
(5)	(PU 。)	.	1:1

Table 1: A derivation composed of only tree-to-string rules for Figure 1.

No.	Rule		
(1)	(IP (NP) (VP (SB) (VP)) (PU))	$X_1\ X_2$	1:1 2:1 3:2 4:2
(2)	(NP (NN 枪手)) (SB 被)	The gunman was	1:1 1:2 2:3
(3)	(VP (NP) (VV 击毙)) (PU 。)	killed by X .	1:3 2:1 3:4
(4)	(NP (NN 警方))	police	1:1

Table 2: A derivation composed of tree-to-string, forest-to-string, and auxiliary rules for Figure 1.

to correspond to one target nonterminal, suggesting that the forest-to-string rules that are rooted at such source nonterminals can be integrated.

For example, Table 2 shows a derivation composed of tree-to-string, forest-to-string, and auxiliary rules for the $\langle T, S, A\rangle$ tuple in Figure 1. r_1 is an auxiliary rule, r_2 and r_3 are forest-to-string rules, and r_4 is a conventional tree-to-string rule.

Following Marcu et al. (2006), we define the probability of a tuple $\langle T, S, A\rangle$ as the sum over all derivations $\theta_i \in \Theta$ that are consistent with the tuple, $c(\Theta) = \langle T, S, A\rangle$. The probability of each derivation θ_i is given by the product of the probabilities of all the rules $p(r_j)$ in the derivation.

$$Pr(T, S, A) = \sum_{\theta_i \in \Theta, c(\Theta)=\langle T,S,A\rangle} \prod_{r_j \in \theta_i} p(r_j) \quad (1)$$

3 Training

We obtain tree-to-string and forest-to-string rules from word-aligned, source side parsed bilingual corpus. The extraction algorithm is shown in Figure 2. Note that T' denotes either a tree or a forest.

For each span, the ⟨tree/forest, string, alignment⟩ triples are identified first. If a triple is consistent with the alignment, the *skeleton* of the triple is computed then. A skeleton s is a rule satisfying the following:

1. $s \in \mathcal{R}(t)$, s is induced from t.
2. $node(T(s)) \geq 2$, the tree/forest of s contains two or more nodes.
3. $\forall r \in \mathcal{R}(t) \wedge node(T(r)) \geq 2, T(s) \subseteq T(r)$, the tree/forest of s is the subgraph of that of any r containing two or more nodes.

Input: a source tree $T = T(f_1^J)$, a target string $S = e_1^I$, and word alignment A between them
$\mathcal{R} := \emptyset$
for $u := 0$ to $J - 1$ **do**
  for $v := 1$ to $J - u$ **do**
    identify the triple set $\mathcal{T}$ corresponding to span $(v, v + u)$
    for each triple $t = \langle T', S', A'\rangle \in \mathcal{T}$ **do**
      if $\langle T', S'\rangle$ is not consistent with A **then**
        continue
      end if
      if $u = 0 \wedge node(T') = 1$ **then**
        add t to $\mathcal{R}$
        add $\langle root(T'), \text{“}X\text{”}, 1{:}1\rangle$ to $\mathcal{R}$
      else
        compute the skeleton s of the triple t
        register rules that are built on s using rules extracted from the sub-triples of t: $\mathcal{R} := \mathcal{R} \cup build(s, \mathcal{R})$
      end if
    end for
  end for
end for
Output: rule set $\mathcal{R}$

Figure 2: Rule extraction algorithm.

Given the skeleton and rules extracted from the sub-triples, the rules for the triple can be acquired.

For example, the algorithm identifies the following triple for span $(1, 2)$ in Figure 1:

⟨(NP (NN 枪手)) (SB 被),“The gunman was”, 1:1 1:2 2:3⟩

The skeleton of the triple is:

⟨(NP) (SB),“$X_1\ X_2$”, 1:1 2:2⟩

As the algorithm proceeds bottom-up, five rules have already been extracted from the sub-triples, rooted at “NP” and “SB” respectively:

⟨(NP),“X”, 1:1⟩
⟨(NP (NN)),“X”, 1:1⟩
⟨(NP (NN 枪手)),“The gunman”, 1:1 1:2⟩

⟨(SB),“X”, 1:1⟩
⟨(SB 被),“was”, 1:1⟩

Hence, we can obtain new rules by replacing the source and target symbols of the skeleton with corresponding rules and also by modifying the alignment information. For the above triple, the combination of the five rules produces $2 \times 3 = 6$ new rules:

⟨(NP) (SB),“X_1 X_2”, 1:1 2:2⟩
⟨(NP) (SB 被),“X was”, 1:1 2:2⟩
⟨((NP (NN)) (SB),“X_1 X_2”, 1:1 2:2⟩
⟨((NP (NN)) (SB 被),“X was”, 1:1 2:2⟩
⟨(NP (NN 枪手)) (SB),“The gunman X”, 1:1 1:2⟩
⟨(NP (NN 枪手)) (SB 被),“The gunman was”, 1:1 1:2 2:3⟩

Since we need only to check the alignment consistency, in principle all phrase pairs can be captured by tree-to-string and forest-to-string rules. To lower the complexity for both training and decoding, we impose four restrictions:

1. Both the first and the last symbols in the target string must be aligned to some source symbols.
2. The height of a tree or forest is no greater than h.
3. The number of direct descendants of a node is no greater than c.
4. The number of leaf nodes is no greater than l.

Although possible, it is infeasible to learn auxiliary rules from training data. To extract an auxiliary rule which integrates at least one forest-to-string rule, one need traverse the parse tree upwards until one reaches a node that subsumes the entire forest without violating the alignment consistency. This usually results in very complex auxiliary rules, especially on real-world training data, making both training and decoding very slow. As a result, we construct auxiliary rules in decoding instead.

4 Decoding

Given a source parse tree $T(f_1^J)$, our decoder finds the target yield of the single best derivation that has source yield of $T(f_1^J)$:

$$\begin{aligned}\hat{S} &= \operatorname*{argmax}_{S,A} Pr(T,S,A) \\ &= \operatorname*{argmax}_{S,A} \sum_{\theta_i \in \Theta, c(\Theta)=\langle T,S,A\rangle} \prod_{r_j \in \theta_i} p(r_j) \\ &\approx \operatorname*{argmax}_{S,A,\theta} \prod_{r_j \in \theta, c(\theta)=\langle T,S,A\rangle} p(r_j) \qquad (2)\end{aligned}$$

Input: a source parse tree $T = T(f_1^J)$
for $u := 0$ to $J-1$ **do**
  for $v := 1$ to $J-u$ **do**
    for each T' spanning from v to $v+u$ **do**
      if T' is a tree **then**
        for each usable tree-to-string rule r **do**
          for each derivation θ inferred from r and derivations in *matrix* **do**
            add θ to *matrix*$[v, v+u, root(T')]$
          end for
        end for
        search subcell divisions $\mathcal{D}[v, v+u]$
        for each subcell division $d \in \mathcal{D}[v, v+u]$ **do**
          if d contains at least one forest cell **then**
            construct auxiliary rule r_a
            for each derivation θ inferred from r_a and derivations in *matrix* **do**
              add θ to *matrix*$[v, v+u, root(T')]$
            end for
          end if
        end for
      else
        for each usable forest-to-string rule r **do**
          for each derivation θ inferred from r and derivations in *matrix* **do**
            add θ to *matrix*$[v, v+u, \text{“”}]$
          end for
        end for
        search subcell divisions $\mathcal{D}[v, v+u]$
      end if
    end for
  end for
end for
find the best derivation $\hat{\theta}$ in *matrix*$[1, J, root(T)]$ and get the best translation $\hat{S} = e(\hat{\theta})$
Output: a target string $\hat{S}$

Figure 3: Decoding algorithm.

Figure 3 demonstrates the decoding algorithm. It organizes the derivations into an array *matrix* whose cells *matrix*$[j_1, j_2, X]$ are sets of derivations. $[j_1, j_2, X]$ represents a tree/forest rooted at X spanning from j_1 to j_2. We use the empty string “” to denote the pseudo root of a forest.

Next, we will explain how to infer derivations for a tree/forest provided a usable rule. If $T(r) = T'$, there is only one derivation which contains only the rule r. This usually happens for leaf nodes. If $T(r) \subset T'$, the rule r resorts to derivations from subcells to infer new derivations. Suppose that the decoder is to translate the source tree in Figure 1 and finds a usable rule for $[1, 5, \text{“IP”}]$:

⟨(IP (NP) (VP) (PU)),“X_1 X_2 X_3”, 1:1 2:2 3:3⟩

Subcell Division	Auxiliary Rule		
$[1,1][2,2][3,5]$	(IP (NP) (VP (SB) (VP)) (PU))	$X_1\ X_2\ X_3$	1:1 2:2 3:3 4:3
$[1,2][3,4][5,5]$	(IP (NP) (VP (SB) (VP)) (PU))	$X_1\ X_2\ X_3$	1:1 2:1 3:2 4:3
$[1,3][4,5]$	(IP (NP) (VP (SB) (VP (NP) (VV))) (PU))	$X_1\ X_2$	1:1 2:1 3:1 4:2 5:2
$[1,1][2,5]$	(IP (NP) (VP) (PU))	$X_1\ X_2$	1:1 2:2 3:2

Table 3: Subcell divisions and corresponding auxiliary rules for the source tree in Figure 1

Since the decoding algorithm proceeds in a bottom-up fashion, the uncovered portions have already been translated.

For $[1, 1, \text{“NP”}]$, suppose that we can find a derivation in *matrix*:

⟨(NP (NN 枪手)),“The gunman”, 1:1 1:2⟩

For $[2, 4, \text{“VP”}]$, we find a derivation in *matrix*:

⟨(VP (SB 被) (VP (NP (NN)) (VV 击毙))),
“was killed by X”, 1:1 2:4 3:2⟩
⟨(NN 警察),“police”, 1:1⟩

For $[5, 5, \text{“PU”}]$, we find a derivation in *matrix*:

⟨(PU 。),“.”, 1:1⟩

Henceforth, we get a derivation for $[1, 5, \text{“IP”}]$, shown in Table 1.

A translation rule r is said to be *usable* to an input tree/forest T' if and only if:

1. $T(r) \subseteq T'$, the tree/forest of r is the subgraph of T'.

2. $root(T(r)) = root(T')$, the root sequence of $T(r)$ is identical to that of T'.

For example, the following rules are usable to the tree “(NP (NR 中国) (NN 经济))”:

⟨(NP (NR) (NN)),“$X_1\ X_2$”, 1:2 2:1⟩
⟨(NP (NR 中国) (NN)),“China X”, 1:1 2:2⟩
⟨(NP (NR 中国) (NN 经济)),“China economy”, 1:1 2:2⟩

Similarly, the forest-to-string rule

⟨((NP (NR) (NN)) (VP)),“$X_1\ X_2\ X_3$”, 1:2 2:1 3:3⟩

is usable to the forest

(NP (NR 布什) (NN 总统)) (VP (VV 发表)(NN 演讲))

As we mentioned before, auxiliary rules are special unlexicalized tree-to-string rules that are built in decoding rather than learnt from real-world data. To get an auxiliary rule for a cell, we need first identify its *subcell division*.

A cell sequence $c_1, c_2, \ldots, c_n$ is referred to as a subcell division of a cell c if and only if:

1. $c_1.begin = c.begin$

2. $c_n.end = c.end$

3. $c_j.end + 1 = c_{j+1}.begin, 1 \leq j < n$

Given a subcell division, it is easy to construct the auxiliary rule for a cell. For each subcell, one need transverse the parse tree upwards until one reaches nodes that subsume it. All descendants of these nodes are dropped. The target string consists of only nonterminals, the number of which is identical to that of subcells. To limit the search space, we assume that the alignment between the source tree and the target string is monotone.

Table 3 shows some subcell divisions and corresponding auxiliary rules constructed for the source tree in Figure 1. For simplicity, we ignore the root node label.

There are 2^{n-1} subcell divisions for a cell which has a length of n. We need only consider the sub-

```
Input: a cell [j1, j2], the derivation array matrix,
  the subcell division array D
if j1 = j2 then
    p̂ := 0
    for each derivation θ in matrix[j1, j2, ·] do
        p̂ := max(p(θ), p̂)
    end for
    add {[j1, j2]} : p̂ to D[j1, j2]
else
    if [j1, j2] is a forest cell then
        p̂ := 0
        for each derivation θ in matrix[j1, j2, ·] do
            p̂ := max(p(θ), p̂)
        end for
        add {[j1, j2]} : p̂ to D[j1, j2]
    end if
    for j := j1 to j2 − 1 do
        for each division d1 ∈ D[j1, j] do
            for each division d2 ∈ D[j + 1, j2] do
                create a new division: d := d1 ⊕ d2
                add d to D[j1, j2]
            end for
        end for
    end for
end if
Output: subcell divisions D[j1, j2]
```

Figure 4: Subcell division search algorithm.

cell divisions which contain at least one forest cell because tree-to-string rules have already explored those contain only tree cells.

The actual search algorithm for subcell divisions is shown in Figure 4. We use *matrix*$[j_1, j_2, \cdot]$ to denote all trees or forests spanning from j_1 to j_2. The subcell divisions and their associated probabilities are stored in an array $\mathcal{D}$. We define an operator $\oplus$ between two divisions: their cell sequences are concatenated and the probabilities are accumulated.

As sometimes there are no usable rules available, we introduce *default* rules to ensure that we can always get a translation for any input parse tree. A default rule is a tree-to-string rule [5], built in two ways:

1. If the input tree contains only one node, the target string of the default rule is equal to the source string.

2. If the height of the input tree is greater than one, the tree of the default rule contains only the root node and its direct descendants of the input tree, the string contains only nonterminals, and the alignment is monotone.

To speed up the decoder, we limit the search space by reducing the number of rules used for each cell. There are two ways to limit the rule table size: by a fixed limit a of how many rules are retrieved for each cell, and by a probability threshold α that specify that the rule probability has to be above some value. Also, instead of keeping the full list of derivations for a cell, we store a top-scoring subset of the derivations. This can also be done by a fixed limit b or a threshold β. The subcell division array $\mathcal{D}$, in which divisions containing forest cells have priority over those composed of only tree cells, is pruned by keeping only a-best divisions.

Following Och and Ney (2002), we base our model on log-linear framework and adopt the seven feature functions described in (Liu et al., 2006). It is very important to balance the preference between conventional tree-to-string rules and the newly-introduced forest-to-string and auxiliary rules. As the probabilities of auxiliary rules are not learnt from training data, we add a feature that sums up the node count of auxiliary rules of a derivation to penalize the use of forest-to-string and auxiliary rules.

[5] There are no default rules for forests because only tree-to-string rules are essential to tree-to-string translation models.

5 Experiments

In this section, we report on experiments with Chinese-to-English translation. The training corpus consists of $31,149$ sentence pairs with $843,256$ Chinese words and $949,583$ English words. For the language model, we used SRI Language Modeling Toolkit (Stolcke, 2002) to train a trigram model with modified Kneser-Ney smoothing (Chen and Goodman, 1998) on the $31,149$ English sentences. We selected 571 short sentences from the 2002 NIST MT Evaluation test set as our development corpus, and used the 2005 NIST MT Evaluation test set as our test corpus. Our evaluation metric is BLEU-4 (Papineni et al., 2002), as calculated by the script mteval-v11b.pl with its default setting except that we used case-sensitive matching of n-grams. To perform minimum error rate training (Och, 2003) to tune the feature weights to maximize the system's BLEU score on development set, we used the script optimizeV5IBMBLEU.m (Venugopal and Vogel, 2005).

We ran GIZA++ (Och and Ney, 2000) on the training corpus in both directions using its default setting, and then applied the refinement rule "diag-and" described in (Koehn et al., 2003) to obtain a single many-to-many word alignment for each sentence pair. Next, we employed a Chinese parser written by Deyi Xiong (Xiong et al., 2005) to parse all the $31,149$ Chinese sentences. The parser was trained on articles 1-270 of Penn Chinese Treebank version 1.0 and achieved 79.4% in terms of F1 measure.

Given the word-aligned, source side parsed bilingual corpus, we obtained bilingual phrases using the training toolkits publicly released by Philipp Koehn with its default setting. Then, we applied extraction algorithm described in Figure 2 to extract both tree-to-string and forest-to-string rules by restricting $h = 3$, $c = 5$, and $l = 7$. All the rules, including bilingual phrases, tree-to-string rules, and forest-to-string rules, are filtered for the development and test sets.

According to different levels of lexicalization, we divide translation rules into three categories:

Rule	L	P	U	Total
BP	251,173	0	0	251,173
TR	56,983	41,027	3,529	101,539
FR	16,609	254,346	25,051	296,006

Table 4: Number of rules used in experiments (BP: bilingual phrase, TR: tree-to-string rule, FR: forest-to-string rule; L: lexicalized, P: partial lexicalized, U: unlexicalized).

System	Rule Set	BLEU4
Pharaoh	BP	0.2182 ± 0.0089
Lynx	BP	0.2059 ± 0.0083
	TR	0.2302 ± 0.0089
	TR + BP	0.2346 ± 0.0088
	TR + FR + AR	0.2402 ± 0.0087

Table 5: Comparison of Pharaoh and Lynx with different rule sets.

1. *lexicalized*: all symbols in both the source and target strings are terminals
2. *unlexicalized*: all symbols in both the source and target strings are nonterminals
3. *partial lexicalized*: otherwise

Table 4 shows the statistics of rules used in our experiments. We find that even though forest-to-string rules are introduced the total number (i.e. $73,592$) of lexicalized tree-to-string and forest-to-string rules is still far less than that (i.e. $251,173$) of bilingual phrases. This difference results from the restriction we impose in training that both the first and last symbols in the target string must be aligned to some source symbols. For the forest-to-string rules, partial lexicalized ones are in the majority.

We compared our system Lynx against a freely available phrase-based decoder Pharaoh (Koehn et al., 2003). For Pharaoh, we set $a = 20$, $\alpha = 0$, $b = 100$, $\beta = 10^{-5}$, and distortion limit $dl = 4$. For Lynx, we set $a = 20$, $\alpha = 0$, $b = 100$, and $\beta = 0$. Two postprocessing procedures ran to improve the outputs of both systems: OOVs removal and recapitalization.

Table 5 shows results on test set using Pharaoh and Lynx with different rule sets. Note that Lynx is capable of using only bilingual phrases plus default rules to perform monotone search. The 95% confidence intervals were computed using Zhang's significance tester (Zhang et al., 2004). We modified it to conform to NIST's current definition of the BLEU brevity penalty. We find that Lynx outperforms Pharaoh significantly. The integration of forest-to-string rules achieves an absolute improvement of 1.0% (4.3% relative) over using tree-to-string rules only. This difference is statistically significant ($p < 0.01$). It also achieves better result than treating bilingual phrases as lexicalized tree-to-string rules. To produce the best result of 0.2402, Lynx made use of $26,082$ tree-to-string rules, $9,219$ default rules, $5,432$ forest-to-string rules, and $2,919$ auxiliary rules. This suggests that tree-to-string rules still play a central role, although the integration of forest-to-string and auxiliary rules is really beneficial.

Forest-to-String Rule Set	BLEU4
None	0.2225 ± 0.0085
L	0.2297 ± 0.0081
P	0.2279 ± 0.0083
U	0.2270 ± 0.0087
L + P + U	0.2312 ± 0.0082

Table 6: Effect of lexicalized, partial lexicalized, and unlexicalized forest-to-string rules.

Table 6 demonstrates the effect of forest-to-string rules with different lexicalization levels. We set $a = 3$, $\alpha = 0$, $b = 10$, and $\beta = 0$. The second row "None" shows the result of using only tree-to-string rules. "L" denotes using tree-to-string rules and lexicalized forest-to-string rules. Similarly, "L+P+U" denotes using tree-to-string rules and all forest-to-string rules. We find that lexicalized forest-to-string rules are more useful.

6 Conclusion

In this paper, we introduce forest-to-string rules to capture non-syntactic phrase pairs that are usually unaccessible to traditional tree-to-string translation models. With the help of auxiliary rules, forest-to-string rules can be integrated into tree-to-string models to offer more general derivations. Experiment results show that the tree-to-string model augmented with forest-to-string rules significantly outperforms

the original model which allows tree-to-string rules only.

Our current rule extraction algorithm attaches the unaligned target words to the nearest ascendants that subsume them. This constraint hampers the expressive power of our model. We will try a more general way as suggested in (Galley et al., 2006), making no a priori assumption about assignment and using EM training to learn the probability distribution. We will also conduct experiments on large scale training data to further examine our design philosophy.

Acknowledgement

This work was supported by National Natural Science Foundation of China, Contract No. 60603095 and 60573188.

References

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical report, Harvard University Center for Research in Computing Technology.

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

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT/NAACL 2004*, pages 273–280, Boston, Massachusetts, USA, May.

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

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT/NAACL 2003*, pages 127–133, Edmonton, Canada, May.

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

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. Spmt: Statistical machine translation with syntactified target language phrases. In *Proceedings of EMNLP 2006*, pages 44–52, Sydney, Australia, July.

Franz J. Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of ACL 2000*, pages 440–447.

Franz J. Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of ACL 2002*, pages 295–302.

Franz J. Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL 2003*, 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 ACL 2002*, pages 311–318, Philadephia, USA, July.

Chris Quirk and Simon Corston-Oliver. 2006. The impact of parse quality on syntactically-informed statistical machine translation. In *Proceedings of EMNLP 2006*, pages 62–69, Sydney, Australia, July.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of ACL 2005*, pages 271–279, Ann Arbor, Michigan, June.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *Proceedings of International Conference on Spoken Language Processing*, volume 30, pages 901–904.

Ashish Venugopal and Stephan Vogel. 2005. Considerations in maximum mutual information and minimum classification error training for statistical machine translation. In *Proceedings of the Tenth Conference of the European Association for Machine Translation*, pages 271–279.

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.

Ying Zhang, Stephan Vogel, and Alex Waibel. 2004. Interpreting bleu/nist scores how much improvement do we need to have a better system? In *Proceedings of Fourth International Conference on Language Resources and Evaluation*, pages 2051–2054.

Ordering Phrases with Function Words

Hendra Setiawan and **Min-Yen Kan**
School of Computing
National University of Singapore
Singapore 117543
{hendrase,kanmy}@comp.nus.edu.sg

Haizhou Li
Institute for Infocomm Research
21 Heng Mui Keng Terrace
Singapore 119613
hli@i2r.a-star.edu.sg

Abstract

This paper presents a Function Word centered, Syntax-based (FWS) solution to address phrase ordering in the context of statistical machine translation (SMT). Motivated by the observation that function words often encode grammatical relationship among phrases within a sentence, we propose a probabilistic synchronous grammar to model the ordering of function words and their left and right arguments. We improve phrase ordering performance by lexicalizing the resulting rules in a small number of cases corresponding to function words. The experiments show that the FWS approach consistently outperforms the baseline system in ordering function words' arguments and improving translation quality in both perfect and noisy word alignment scenarios.

1 Introduction

The focus of this paper is on function words, a class of words with little intrinsic meaning but is vital in expressing grammatical relationships among words within a sentence. Such encoded grammatical information, often implicit, makes function words pivotal in modeling structural divergences, as projecting them in different languages often result in long-range structural changes to the realized sentences.

Just as a foreign language learner often makes mistakes in using function words, we observe that current machine translation (MT) systems often perform poorly in ordering function words' arguments; lexically correct translations often end up reordered incorrectly. Thus, we are interested in modeling the structural divergence encoded by such function words. A key finding of our work is that modeling the ordering of the dependent arguments of function words results in better translation quality.

Most current systems use statistical knowledge obtained from corpora in favor of rich natural language knowledge. Instead of using syntactic knowledge to determine function words, we approximate this by equating the most frequent words as function words. By explicitly modeling phrase ordering around these frequent words, we aim to capture the most important and prevalent ordering productions.

2 Related Work

A good translation should be both faithful with adequate lexical choice to the source language and fluent in its word ordering to the target language. In pursuit of better translation, phrase-based models (Och and Ney, 2004) have significantly improved the quality over classical word-based models (Brown et al., 1993). These multiword phrasal units contribute to fluency by inherently capturing intra-phrase reordering. However, despite this progress, inter-phrase reordering (especially long distance ones) still poses a great challenge to statistical machine translation (SMT).

The basic phrase reordering model is a simple unlexicalized, context-insensitive distortion penalty model (Koehn et al., 2003). This model assumes little or no structural divergence between language pairs, preferring the original, translated order by penalizing reordering. This simple model works well when properly coupled with a well-trained language

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 712–719,
Prague, Czech Republic, June 2007.

model, but is otherwise impoverished without any lexical evidence to characterize the reordering.

To address this, lexicalized context-sensitive models incorporate contextual evidence. The local prediction model (Tillmann and Zhang, 2005) models structural divergence as the relative position between the translation of two neighboring phrases. Other further generalizations of orientation include the global prediction model (Nagata et al., 2006) and distortion model (Al-Onaizan and Papineni, 2006).

However, these models are often fully lexicalized and sensitive to individual phrases. As a result, they are not robust to unseen phrases. A careful approximation is vital to avoid data sparseness. Proposals to alleviate this problem include utilizing bilingual phrase cluster or words at the phrase boundary (Nagata et al., 2006) as the phrase identity.

The benefit of introducing lexical evidence without being fully lexicalized has been demonstrated by a recent state-of-the-art *formally* syntax-based model[1], Hiero (Chiang, 2005). Hiero performs phrase ordering by using linked non-terminal symbols in its synchronous CFG production rules coupled with lexical evidence. However, since it is difficult to specify a well-defined rule, Hiero has to rely on weak heuristics (i.e., length-based thresholds) to extract rules. As a result, Hiero produces grammars of enormous size. Watanabe et al. (2006) further reduces the grammar's size by enforcing all rules to comply with Greibach Normal Form.

Taking the lexicalization an intuitive a step forward, we propose a novel, finer-grained solution which models the content and context information encoded by function words - approximated by high frequency words. Inspired by the success of syntax-based approaches, we propose a synchronous grammar that accommodates gapping production rules, while focusing on the statistical modeling in relation to function words. We refer to our approach as the Function Word-centered Syntax-based approach (FWS). Our FWS approach is different from Hiero in two key aspects. First, we use only a small set of high frequency lexical items to lexicalize non-terminals in the grammar. This results in a much smaller set of rules compared to Hiero,

[1]Chiang (2005) used the term "formal" to indicate the use of synchronous grammar but without linguistic commitment

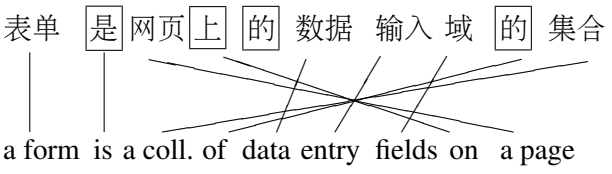


Figure 1: A Chinese-English sentence pair.

greatly reducing the computational overhead that arises when moving from phrase-based to syntax-based approach. Furthermore, by modeling only high frequency words, we are able to obtain reliable statistics even in small datasets. Second, as opposed to Hiero, where phrase ordering is done implicitly alongside phrase translation and lexical weighting, we directly model the reordering process using orientation statistics.

The FWS approach is also akin to (Xiong et al., 2006) in using a synchronous grammar as a reordering constraint. Instead of using Inversion Transduction Grammar (ITG) (Wu, 1997) directly, we will discuss an ITG extension to accommodate gapping.

3 Phrase Ordering around Function Words

We use the following Chinese (c) to English (e) translation in Fig.1 as an illustration to conduct an inquiry to the problem. Note that the sentence translation requires some translations of English words to be ordered far from their original position in Chinese. Recovering the correct English ordering requires the inversion of the Chinese postpositional phrase, followed by the inversion of the first smaller noun phrase, and finally the inversion of the second larger noun phrase. Nevertheless, the correct ordering can be recovered if the position and the semantic roles of the arguments of the boxed function words were known. Such a function word centered approach also hinges on knowing the correct phrase boundaries for the function words' arguments and which reorderings are given precedence, in case of conflicts.

We propose modeling these sources of knowledge using a statistical formalism. It includes 1) a model to capture bilingual orientations of the left and right arguments of these function words; 2) a model to approximate correct reordering sequence; and 3) a model for finding constituent boundaries of

the left and right arguments. Assuming that the most frequent words in a language are function words, we can apply orientation statistics associated with these words to reorder their adjacent left and right neighbors. We follow the notation in (Nagata et al., 2006) and define the following bilingual orientation values given two neighboring source (Chinese) phrases: Monotone-Adjacent (MA); Reverse-Adjacent (RA); Monotone-Gap (MG); and Reverse-Gap (RG). The first clause (monotone, reverse) indicates whether the target language translation order follows the source order; the second (adjacent, gap) indicates whether the source phrases are adjacent or separated by an intervening phrase on the target side.

Table 1 shows the orientation statistics for several function words. Note that we separate the statistics for left and right arguments to account for differences in argument structures: some function words take a single argument (e.g., prepositions), while others take two or more (e.g., copulas). To handle other reordering decisions not explicitly encoded (i.e., lexicalized) in our FWS model, we introduce a universal token $\mathcal{U}$, to be used as a backoff statistic when function words are absent.

For example, orientation statistics for 是 (to be) overwhelmingly suggests that the English translation of its surrounding phrases is identical to its Chinese ordering. This reflects the fact that the arguments of copulas in both languages are realized in the same order. The orientation statistics for postposition 上 (on) suggests inversion which captures the divergence between Chinese postposition to the English preposition. Similarly, the dominant orientation for particle 的 (of) suggests the noun-phrase shift from modified-modifier to modifier-modified, which is common when translating Chinese noun phrases to English.

Taking all parts of the model, which we detail later, together with the knowledge in Table 1, we demonstrate the steps taken to translate the example in Fig. 2. We highlight the function words with boxed characters and encapsulate content words as indexed symbols. As shown, orientation statistics from function words alone are adequate to recover the English ordering - in practice, content words also influence the reordering through a language model. One can think of the FWS approach as a foreign language learner with limited knowledge about Chinese grammar but fairly knowledgable about the role of Chinese function words.

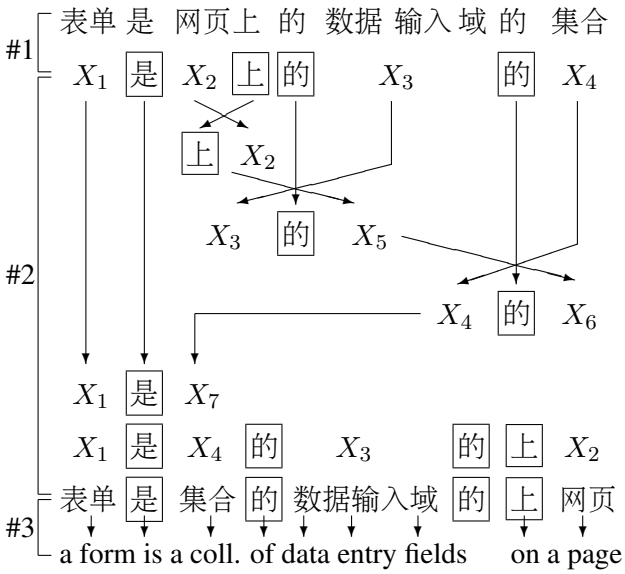


Figure 2: In Step 1, function words (boxed characters) and content words (indexed symbols) are identified. Step 2 reorders phrases according to knowledge embedded in function words. A new indexed symbol is introduced to indicate previously reordered phrases for conciseness. Step 3 finally maps Chinese phrases to their English translation.

4 The FWS Model

We first discuss the extension of standard ITG to accommodate gapping and then detail the statistical components of the model later.

4.1 Single Gap ITG (SG-ITG)

The FWS model employs a synchronous grammar to describe the admissible orderings.

The utility of ITG as a reordering constraint for most language pairs, is well-known both empirically (Zens and Ney, 2003) and analytically (Wu, 1997), however ITG's *straight* (monotone) and *inverted* (reverse) rules exhibit strong cohesiveness, which is inadequate to express orientations that require gaps. We propose SG-ITG that follows Wellington et al. (2006)'s suggestion to model at most one gap.

We show the rules for SG-ITG below. Rules 1-3 are identical to those defined in standard ITG, in which monotone and reverse orderings are represented by square and angle brackets, respectively.

Rank	Word	unigram	MA_L	RA_L	MG_L	RG_L	MA_R	RA_R	MG_R	RG_R
1	的	0.0580	0.45	**0.52**	0.01	0.02	0.44	**0.52**	0.01	0.03
2	,	0.0507	**0.85**	0.12	0.02	0.01	**0.84**	0.12	0.02	0.02
3	。	0.0550	**0.99**	0.01	0.00	0.00	**0.92**	0.08	0.00	0.00
4	”	0.0155	**0.87**	0.10	0.02	0.00	**0.82**	0.12	0.05	0.02
5	“	0.0153	**0.84**	0.11	0.01	0.04	**0.88**	0.11	0.01	0.01
6	和	0.0138	**0.95**	0.02	0.01	0.01	**0.97**	0.02	0.01	0.00
7	任务	0.0123	**0.73**	0.12	0.10	0.04	**0.51**	0.14	0.14	0.20
8	可以	0.0114	**0.78**	0.12	0.03	0.07	**0.86**	0.05	0.08	0.01
9	或	0.0099	**0.95**	0.02	0.02	0.01	**0.96**	0.01	0.02	0.01
10	将	0.0091	**0.87**	0.10	0.01	0.02	**0.88**	0.10	0.01	0.00
21	是	0.0056	**0.85**	0.11	0.02	0.02	**0.85**	0.04	0.09	0.02
37	上	0.0035	0.33	**0.65**	0.02	0.01	0.31	**0.63**	0.03	0.03
-	$\mathcal{U}$	0.0002	**0.76**	0.14	0.06	0.05	**0.74**	0.13	0.07	0.06

Table 1: Orientation statistics and unigram probability of selected frequent Chinese words in the HIT corpus. Subscripts L/R refers to lexical unit's orientation with respect to its left/right neighbor. $\mathcal{U}$ is the universal token used in back-off for $N = 128$. Dominant orientations of each word are in **bold**.

(1) $X \rightarrow c/e$
(2) $X \rightarrow [XX]$ (3) $X \rightarrow \langle XX \rangle$
(4) $X^{\diamond} \rightarrow [X \diamond X]$ (5) $X^{\diamond} \rightarrow \langle X \diamond X \rangle$
(6) $X \rightarrow [X * X]$ (7) $X \rightarrow \langle X * X \rangle$

SG-ITG introduces two new sets of rules: gapping (Rules 4-5) and dovetailing (Rules 6-7) that deal specifically with gaps. On the RHS of the gapping rules, a diamond symbol ($\diamond$) indicates a gap, while on the LHS, it emits a superscripted symbol $X^{\diamond}$ to indicate a gapped phrase (plain Xs without superscripts are thus contiguous phrases). Gaps in $X^{\diamond}$ are eventually filled by actual phrases via dovetailing (marked with an $*$ on the RHS).

Fig.3 illustrates gapping and dovetailing rules using an example where two Chinese adjectival phrases are translated into a single English subordinate clause. SG-ITG can generate the correct ordering by employing gapping followed by dovetailing, as shown in the following simplified trace:

$X_1^{\diamond}$ $\rightarrow$ ⟨ 1997 的 第一版, **V.1** $\diamond$ **1997** ⟩
$X_2^{\diamond}$ $\rightarrow$ ⟨ 1998 的 第二版, **V.2** $\diamond$ **1998** ⟩
X_3 $\rightarrow$ $[X_1 * X_2]$
$\rightarrow$ [1997 的 第一版 和 1998 的 第二版,
V.1 $\diamond$ **1997** $*$ **V.2** $\diamond$ **1998**]
$\rightarrow$ 1997 的第一版和1998 的第二版,
V.1 and **V.2** that were released in **1997** and **1998**

where $X_1^{\diamond}$ and $X_2^{\diamond}$ each generate the translation of their respective Chinese noun phrase using gapping and X_3 generates the English subclause by dovetailing the two gapped phrases together.

Thus far, the grammar is unlexicalized, and does not incorporate any lexical evidence. Now we modify the grammar to introduce lexicalized function words to SG-ITG. In practice, we introduce a new set of lexicalized non-terminal symbols Y_i, $i \in \{1...N\}$, to represent the top N most-frequent words in the vocabulary; the existing unlexicalized X is now reserved for content words. This difference does not inherently affect the structure of the grammar, but rather lexicalizes the statistical model.

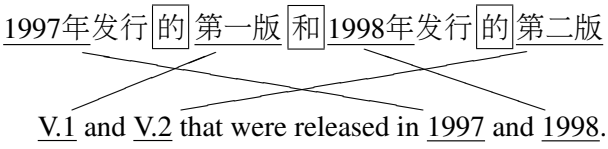


Figure 3: An example of an alignment that can be generated only by allowing gaps.

In this way, although different Y_is follow the same production rules, they are associated with different statistics. This is reflected in Rules 8-9. Rule 8 emits the function word; Rule 9 reorders the arguments around the function word, resembling our orientation model (see Section 4.2) where a function word influences the orientation of its left and right arguments. For clarity, we omit notation that denotes which rules have been applied (monotone, reverse; gapping, dovetailing).

(8) $Y_i \rightarrow c/e$ (9) $X \rightarrow XY_iX$

In practice, we replace Rule 9 with its equivalent 2-normal form set of rules (Rules 10-13). Finally, we introduce rules to handle back-off (Rules 14-16) and upgrade (Rule 17). These allow SG-ITG to re-

vert function words to normal words and vice versa.

(10) $R \rightarrow Y_i X$ (11) $L \rightarrow X Y_i$
(12) $X \rightarrow L X$ (13) $X \rightarrow X R$
(14) $Y_i \rightarrow X$ (15) $R \rightarrow X$
(16) $L \rightarrow X$ (17) $X \rightarrow Y_{\mathcal{U}}$

Back-off rules are needed when the grammar has to reorder two adjacent function words, where one set of orientation statistics must take precedence over the other. The example in Fig.1 illustrates such a case where the orientation of 上 (on) and 的 (of) compete for influence. In this case, the grammar chooses to use 的 (of) and reverts the function word 上 (on) to the unlexicalized form.

The upgrade rule is used for cases where there are two adjacent phrases, both of which are not function words. Upgrading allows either phrase to act as a function word, making use of the universal word's orientation statistics to reorder its neighbor.

4.2 Statistical model

We now formulate the FWS model as a statistical framework. We replace the deterministic rules in our SG-ITG grammar with probabilistic ones, elevating it to a stochastic grammar. In particular, we develop the three sub models (see Section 3) which influence the choice of production rules for ordering decision. These models operate on the 2-norm rules, where the RHS contains one function word and its argument (except in the case of the phrase boundary model). We provide the intuition for these models next, but their actual form will be discussed in the next section on training.

1) Orientation Model *ori(o|H,Y_i)***:** This model captures the preference of a function word Y_i to a particular orientation $o \in \{MA, RA, MG, RG\}$ in reordering its $H \in \{left, right\}$ argument X. The parameter H determines which set of Y_i's statistics to use (left or right); the model consults Y_i's left orientation statistic for Rules 11 and 13 where X precedes Y_i, otherwise Y_i's right orientation statistic is used for Rules 10 and 12.

2) Preference Model *pref(Y_i)***:** This model arbitrates reordering in the cases where two function words are adjacent and the backoff rules have to decide which function word takes precedence, reverting the other to the unlexicalized X form. This model prefers the function word with higher unigram probability to take the precedence.

3) Phrase Boundary Model *pb(X)***:** This model is a penalty-based model, favoring the resulting alignment that conforms to the source constituent boundary. It penalizes Rule 1 if the terminal rule X emits a Chinese phrase that violates the boundary ($pb = e^{-1}$), otherwise it is inactive ($pb = 1$).

These three sub models act as features alongside seven other standard SMT features in a log-linear model, resulting in the following set of features $\{f_1, \ldots, f_{10}\}$: f_1) orientation $ori(o|H, Y_i)$; f_2) preference $pref(Y_i)$; f_3) phrase boundary $pb(X)$; f_4) language model $lm(e)$; $f_5 - f_6$) phrase translation score $\phi(e|c)$ and its inverse $\phi(c|e)$; $f_7 - f_8$) lexical weight $lex(e|c)$ and its inverse $lex(c|e)$; f_9) word penalty wp; and f_{10}) phrase penalty pp.

The translation is then obtained from the most probable derivation of the stochastic SG-ITG. The formula for a single derivation is shown in Eq. (18), where $X_1, X_2, ..., X_L$ is a sequence of rules with $w(X_l)$ being the weight of each particular rule X_l. $w(X_l)$ is estimated through a log-linear model, as in Eq. (19), with all the abovementioned features where λ_j reflects the contribution of each feature f_j.

$$(18) \quad P(X_1, ..., X_L) = \prod_{l=1}^{L} w(X_l)$$

$$(19) \quad w(X_l) = \prod_{j=1}^{10} f_j(X_l)^{\lambda_j}$$

5 Training

We train the orientation and preference models from statistics of a training corpus. To this end, we first derive the event counts and then compute the relative frequency of each event. The remaining phrase boundary model can be modeled by the output of a standard text chunker, as in practice it is simply a constituent boundary detection mechanism together with a penalty scheme.

The events of interest to the orientation model are (Y_i, o) tuples, where $o \in \{MA, RA, MG, RG\}$ is an orientation value of a particular function word Y_i. Note that these tuples are not directly observable from training data. Hence, we need an algorithm to derive (Y_i, o) tuples from a parallel corpus. Since both left and right statistics share identical training steps, thus we omit references to them.

The algorithm to derive (Y_i, o) involves several steps. First, we estimate the bi-directional alignment

by running GIZA++ and applying the "grow-diag-final" heuristic. Then, the algorithm enumerates all Y_i and determines its orientation o with respect to its argument X to derive (Y_i, o). To determine o, the algorithm inspects the monotonicity (monotone or reverse) and adjacency (adjacent or gap) between Y_i's and X's alignments.

Monotonicity can be determined by looking at the Y_i's alignment with respect to the most fine-grained level of X (i.e., word level alignment). However, such a heuristic may inaccurately suggest gap orientation. Figure 1 illustrates this problem when deriving the orientation for the second 的 (of). Looking only at the word alignment of its left argument 域 (fields) incorrectly suggests a gapped orientation, where the alignment of 数据输入 (data entry) intervened. It is desirable to look at the alignment of 数据输入域 (data entry fields) at the phrase level, which suggests the correct adjacent orientation instead.

To address this issue, the algorithm uses gapping conservatively by utilizing the consistency constraint (Och and Ney, 2004) to suggest phrase level alignment of X. The algorithm exhaustively grows consistent blocks containing the most fine-grained level of X not including Y_i. Subsequently, it merges each hypothetical argument with the Y_i's alignment. The algorithm decides that Y_i has a gapped orientation only if all merged blocks violate the consistency constraint, concluding an adjacent orientation otherwise.

With the event counts $C(Y_i, o)$ of tuple (Y_i, o), we estimate the orientation model for Y_i and $\mathcal{U}$ using Eqs. (20) and (21). We also estimate the preference model with word unigram counts $C(Y_i)$ using Eqs. (22) and (23), where V indicates the vocabulary size.

$$(20)\ ori(o|Y_i) = C(Y_i, o)/C(Y_i, \cdot), i \leqslant N$$

$$(21)\ ori(o|\mathcal{U}) = \sum_{i>N} C(Y_i, o) / \sum_{i>N} C(Y_i, \cdot)$$

$$(22)\ pref(Y_i) = C(Y_i)/C(\cdot), i \leqslant N$$

$$(23)\ pref(\mathcal{U}) = 1/(V-N) \sum_{i>N} C(Y_i)/C(\cdot)$$

Samples of these statistics are found in Table 1 and have been used in the running examples. For instance, the statistic $ori(RA_L|的) = 0.52$, which is the dominant one, suggests that the grammar inversely order 的(of)'s left argument; while in our illustration of backoff rules in Fig.1, the grammar chooses 的(of) to take precedence since $pref(的) > pref(上)$.

6 Decoding

We employ a bottom-up CKY parser with a beam to find the derivation of a Chinese sentence which maximizes Eq. (18). The English translation is then obtained by post-processing the best parse.

We set the beam size to 30 in our experiment and further constrain reordering to occur within a window of 10 words. Our decoder also prunes entries that violate the following constraints: 1) each entry contains at most one gap; 2) any gapped entries must be dovetailed at the next level higher; 3) an entry spanning the whole sentence must not contain gaps.

The score of each newly-created entry is derived from the scores of its parts accordingly. When scoring entries, we treat gapped entries as contiguous phrases by ignoring the gap symbol and rely on the orientation model to penalize such entries. This allows a fair score comparison between gapped and contiguous entries.

7 Experiments

We would like to study how the FWS model affects 1) the ordering of phrases around function words; 2) the overall translation quality. We achieve this by evaluating the FWS model against a baseline system using two metrics, namely, orientation accuracy and BLEU respectively.

We define the orientation accuracy of a (function) word as the accuracy of assigning correct orientation values to both its left and right arguments. We report the aggregate for the top 1024 most frequent words; these words cover 90% of the test set.

We devise a series of experiments and run it in two scenarios - manual and automatic alignment - to assess the effects of using perfect or real-world input. We utilize the HIT bilingual computer manual corpus, which has been manually aligned, to perform Chinese-to-English translation (see Table 2). Manual alignment is essential as we need to measure orientation accuracy with respect to a gold standard.

		Chinese	English
train	words	145,731	135,032
(7K sentences)	vocabulary	5,267	8,064
dev	words	13,986	14,638
(1K sentences)	untranslatable	486 (3.47%)	
test	words	27,732	28,490
(2K sentences)	untranslatable	935 (3.37%)	

Table 2: Statistics for the HIT corpus.

A language model is trained using the SRILM-Toolkit, and a text chunker (Chen et al., 2006) is applied to the Chinese sentences in the test and dev sets to extract the constituent boundaries necessary for the phrase boundary model. We run minimum error rate training on dev set using Chiang's toolkit to find a set of parameters that optimizes BLEU score.

7.1 Perfect Lexical Choice

Here, the task is simplified to recovering the correct order of the English sentence from the scrambled Chinese order. We trained the orientation model using manual alignment as input. The aforementioned decoder is used with phrase translation, lexical mapping and penalty features turned off.

Table 4 compares orientation accuracy and BLEU between our FWS model and the baseline. The baseline (lm+d) employs a language model and distortion penalty features, emulating the standard Pharaoh model. We study the behavior of the FWS model with different numbers of lexicalized items N. We start with the language model alone (N=0) and incrementally add the orientation (+ori), preference (+ori+pref) and phrase boundary models (+ori+pref+pb).

As shown, the language model alone is relatively weak, assigning the correct orientation in only 62.28% of the cases. A closer inspection reveals that the lm component aggressively promotes reverse reorderings. Including a distortion penalty model (the baseline) improves the accuracy to 72.55%. This trend is also apparent for the BLEU score.

When we incorporate the FSW model, including just the most frequent word (Y_1=的), we see improvement. This model promotes non-monotone reordering conservatively around Y_1 (where the dominant statistic suggests reverse ordering). Increasing the value of N leads to greater improvement. The most effective improvement is obtained by increasing N to 128. Additional (marginal) improvement is obtained at the expense of modeling an additional 900+ lexical items. We see these results as validating our claim that modeling the top few most frequent words captures most important and prevalent ordering productions.

pharaoh (dl=5)	22.44 ± 0.94
+ori	23.80 ± 0.98
+ori+pref	23.85 ± 1.00
+ori+pref+pb	23.86 ± 1.08

Table 3: BLEU score with the 95% confidence intervals based on (Zhang and Vogel, 2004). All improvement over the baseline (row 1) are statistically significant under paired bootstrap resampling.

Lastly, we study the effect of the pref and pb features. The inclusion of both sub models has little affect on orientation accuracy, but it improves BLEU consistently (although not significantly). This suggests that both models correct the mistakes made by the ori model while preserving the gain. They are not as effective as the addition of the basic orientation model as they only play a role when two lexicalized entries are adjacent.

7.2 Full SMT experiments

Here, all knowledge is automatically trained on the train set, and as a result, the input word alignment is noisy. As a baseline, we use the state-of-the-art phrase-based Pharaoh decoder. For a fair comparison, we run minimum error rate training for different distortion limits from 0 to 10 and report the best parameter (dl=5) as the baseline.

We use the phrase translation table from the baseline and perform an identical set of experiments as the perfect lexical choice scenario, except that we only report the result for N=128, due to space constraint. Table 3 reports the resulting BLEU scores.

As shown, the FWS model improves BLEU score significantly over the baseline. We observe the same trend as the one in perfect lexical choice scenario where top 128 most frequent words provides the majority of improvement. However, the pb features yields no noticeable improvement unlike in prefect lexical choice scenario; this is similar to the findings in (Koehn et al., 2003).

		N=0	N=1	N=4	N=16	N=64	N=128	N=256	N=1024
Orientation Acc. (%)	lm+d	72.55							
	+ori	62.28	76.52	76.58	77.38	77.54	**78.17**	77.76	*78.38*
	+ori+pref		76.66	76.82	77.57	77.74	**78.13**	77.94	*78.54*
	+ori+pref+pb		76.70	76.85	77.58	77.70	**78.20**	77.94	*78.56*
BLEU	lm+d	75.13							
	+ori	66.54	77.54	77.57	78.22	78.48	**78.76**	78.58	*79.20*
	+ori+pref		77.60	77.70	78.29	78.65	**78.77**	78.70	*79.30*
	+ori+pref+pb		77.69	77.80	78.34	78.65	**78.93**	78.79	*79.30*

Table 4: Results using perfect aligned input. Here, (lm+d) is the baseline; (+ori), (+ori+pref) and (+ori+pref+pb) are different FWS configurations. The results of the model (where N is varied) that features the largest gain are **bold**, whereas the highest score is *italicized*.

8 Conclusion

In this paper, we present a statistical model to capture the grammatical information encoded in function words. Formally, we develop the Function Word Syntax-based (FWS) model, a probabilistic synchronous grammar, to encode the orientation statistics of arguments to function words. Our experimental results shows that the FWS model significantly improves the state-of-the-art phrase-based model.

We have touched only the surface benefits of modeling function words. In particular, our proposal is limited to modeling function words in the source language. We believe that conditioning on both source and target pair would result in more fine-grained, accurate orientation statistics.

From our error analysis, we observe that 1) reordering may span several levels and the preference model does not handle this phenomena well; 2) correctly reordered phrases with incorrect boundaries severely affects BLEU score and the phrase boundary model is inadequate to correct the boundaries especially for cases of long phrase. In future, we hope to address these issues while maintaining the benefits offered by modeling function words.

References

Benjamin Wellington, Sonjia Waxmonsky, and I. Dan Melamed. 2006. Empirical Lower Bounds on the Complexity of Translational Equivalence. In *ACL/COLING 2006*, pp. 977–984.

Christoph Tillman and Tong Zhang. 2005. A Localized Prediction Model for Statistical Machine Translation. In *ACL 2005*, pp. 557–564.

David Chiang. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation. In *ACL 2005*, pp. 263–270.

Dekai Wu. 1997. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, 23(3):377–403.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation. In *ACL/COLING 2006*, pp. 521–528.

Franz J. Och and Hermann Ney. 2004. The Alignment Template Approach to Statistical Machine Translation. *Computational Linguistics*, 30(4):417–449.

Masaaki Nagata, Kuniko Saito, Kazuhide Yamamoto, and Kazuteru Ohashi. 2006. A Clustered Global Phrase Reordering Model for Statistical Machine Translation. In *ACL/COLING 2006*, pp. 713–720.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, Robert L. Mercer 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *HLT-NAACL 2003*, pp. 127–133.

Richard Zens and Hermann Ney. 2003. A Comparative Study on Reordering Constraints in Statistical Machine Translation. In *ACL 2003*.

Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. 2006. Left-to-Right Target Generation for Hierarchical Phrase-Based Translation. In *ACL/COLING 2006*, pp. 777–784.

Wenliang Chen, Yujie Zhang and Hitoshi Isahara 2006. An Empirical Study of Chinese Chunking In *ACL 2006 Poster Sessions*, pp. 97–104.

Yaser Al-Onaizan and Kishore Papineni. 2006. Distortion Models for Statistical Machine Translation. In *ACL/COLING 2006*, pp. 529–536.

Ying Zhang and Stephan Vogel. 2004. Measuring Confidence Intervals for the Machine Translation Evaluation Metrics. In *TMI 2004*.

A Probabilistic Approach to Syntax-based Reordering for Statistical Machine Translation

Chi-Ho Li, Dongdong Zhang, Mu Li, Ming Zhou
Microsoft Research Asia
Beijing, China
chl, dozhang@microsoft.com
muli, mingzhou@microsoft.com

Minghui Li, Yi Guan
Harbin Institute of Technology
Harbin, China
mhli@insun.hit.edu.cn
guanyi@insun.hit.edu.cn

Abstract

Inspired by previous preprocessing approaches to SMT, this paper proposes a novel, probabilistic approach to reordering which combines the merits of syntax and phrase-based SMT. Given a source sentence and its parse tree, our method generates, by tree operations, an n-best list of reordered inputs, which are then fed to standard phrase-based decoder to produce the optimal translation. Experiments show that, for the NIST MT-05 task of Chinese-to-English translation, the proposal leads to BLEU improvement of 1.56%.

1 Introduction

The phrase-based approach has been considered the default strategy to Statistical Machine Translation (SMT) in recent years. It is widely known that the phrase-based approach is powerful in local lexical choice and word reordering within short distance. However, long-distance reordering is problematic in phrase-based SMT. For example, the distance-based reordering model (Koehn et al., 2003) allows a decoder to translate in non-monotonous order, under the constraint that the distance between two phrases translated consecutively does not exceed a limit known as *distortion limit*. In theory the distortion limit can be assigned a very large value so that all possible reorderings are allowed, yet in practise it is observed that too high a distortion limit not only harms efficiency but also translation performance (Koehn et al., 2005). In our own experiment setting, the best distortion limit for Chinese-English translation is 4. However, some ideal translations exhibit reorderings longer than such distortion limit. Consider the sentence pair in NIST MT-2005 test set shown in figure 1(a): after translating the word "修补/*mend*", the decoder should 'jump' across six words and translate the last phrase "关系 裂缝/*fissures in the relationship*". Therefore, while short-distance reordering is under the scope of the distance-based model, long-distance reordering is simply out of the question.

A terminological remark: In the rest of the paper, we will use the terms *global reordering* and *local reordering* in place of long-distance reordering and short-distance reordering respectively. The distinction between long and short distance reordering is solely defined by distortion limit.

Syntax[1] is certainly a potential solution to global reordering. For example, for the last two Chinese phrases in figure 1(a), simply swapping the two children of the NP node will produce the correct word order on the English side. However, there are also reorderings which do not agree with syntactic analysis. Figure 1(b) shows how our phrase-based decoder[2] obtains a good English translation by reordering two blocks. It should be noted that the second Chinese block "结束 时" and its English counterpart "*at the end of*" are not constituents at all.

In this paper, our interest is the value of syntax in reordering, and the major statement is that syntactic information is useful in handling global reordering

[1] Here by syntax it is meant linguistic syntax rather than formal syntax.

[2] The decoder is introduced in section 6.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 720–727,
Prague, Czech Republic, June 2007.

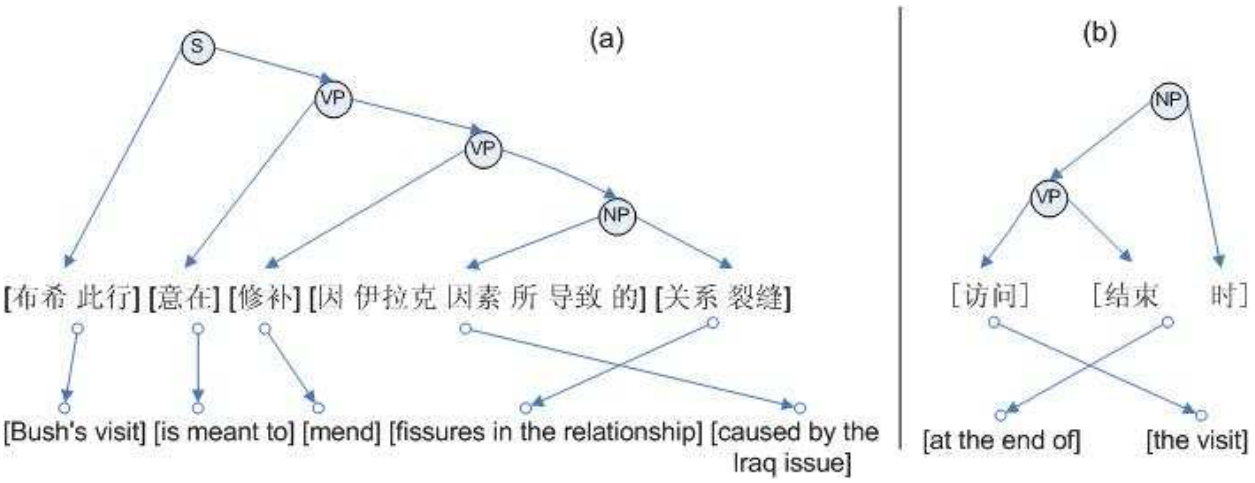


Figure 1: Examples on how syntax (a) helps and (b) harms reordering in Chinese-to-English translation
The lines and nodes on the top half of the figures show the phrase structure of the Chinese sentences, while the links on the bottom half of the figures show the alignments between Chinese and English phrases. Square brackets indicate the boundaries of blocks found by our decoder.

and it achieves better MT performance on the basis of the standard phrase-based model. To prove it, we developed a hybrid approach which preserves the strength of phrase-based SMT in local reordering as well as the strength of syntax in global reordering.

Our method is inspired by previous *preprocessing* approaches like (Xia and McCord, 2004), (Collins et al., 2005), and (Costa-jussà and Fonollosa, 2006), which split translation into two stages:

$$S \rightarrow S' \rightarrow T \quad (1)$$

where a sentence of the source language (SL), S, is first reordered with respect to the word order of the target language (TL), and then the reordered SL sentence S' is translated as a TL sentence T by monotonous translation.

Our first contribution is a new translation model as represented by formula 2:

$$S \rightarrow n \times S' \rightarrow n \times T \rightarrow \hat{T} \quad (2)$$

where an n-best list of S', instead of only one S', is generated. The reason of such change will be given in section 2. Note also that the translation process $S' \rightarrow T$ is not monotonous, since the distance-based model is needed for local reordering. Our second contribution is our definition of the best translation:

$$\arg\max_{T} \exp(\lambda_r \log Pr(S \rightarrow S') + \sum_i \lambda_i F_i(S' \rightarrow T))$$

where F_i are the features in the standard phrase-based model and $Pr(S \rightarrow S')$ is our new feature, viz. the probability of reordering S as S'. The details of this model are elaborated in sections 3 to 6. The settings and results of experiments on this new model are given in section 7.

2 Related Work

There have been various attempts to syntax-based SMT, such as (Yamada and Knight, 2001) and (Quirk et al., 2005). We do not adopt these models since a lot of subtle issues would then be introduced due to the complexity of syntax-based decoder, and the impact of syntax on reordering will be difficult to single out.

There have been many reordering strategies under the phrase-based camp. A notable approach is lexicalized reordering (Koehn et al., 2005) and (Tillmann, 2004). It should be noted that this approach achieves the best result within certain distortion limit and is therefore not a good model for global reordering.

There are a few attempts to the preprocessing approach to reordering. The most notable ones are (Xia and McCord, 2004) and (Collins et al., 2005), both of which make use of linguistic syntax in the preprocessing stage. (Collins et al., 2005) analyze German clause structure and propose six types

of rules for transforming German parse trees with respect to English word order. Instead of relying on manual rules, (Xia and McCord, 2004) propose a method in learning patterns of rewriting SL sentences. This method parses training data and uses some heuristics to align SL phrases with TL ones. From such alignment it can extract rewriting patterns, of which the units are words and POSs. The learned rewriting rules are then applied to rewrite SL sentences before monotonous translation.

Despite the encouraging results reported in these papers, the two attempts share the same shortcoming that their reordering is deterministic. As pointed out in (Al-Onaizan and Papineni, 2006), these strategies make hard decisions in reordering which cannot be undone during decoding. That is, the choice of reordering is independent from other translation factors, and once a reordering mistake is made, it cannot be corrected by the subsequent decoding.

To overcome this weakness, we suggest a method to 'soften' the hard decisions in preprocessing. The essence is that our preprocessing module generates n-best S's rather than merely one S'. A variety of reordered SL sentences are fed to the decoder so that the decoder can consider, to certain extent, the interaction between reordering and other factors of translation. The entire process can be depicted by formula 2, recapitulated as follows:

$$S \rightarrow n \times S' \rightarrow n \times T \rightarrow \hat{T}.$$

Apart from their deterministic nature, the two previous preprocessing approaches have their own weaknesses. (Collins et al., 2005) count on manual rules and it is suspicious if reordering rules for other language pairs can be easily made. (Xia and McCord, 2004) propose a way to learn rewriting patterns, nevertheless the units of such patterns are words and their POSs. Although there is no limit to the length of rewriting patterns, due to data sparseness most patterns being applied would be short ones. Many instances of global reordering are therefore left unhandled.

3 The Acquisition of Reordering Knowledge

To avoid this problem, we give up using rewriting patterns and design a form of reordering knowledge which can be directly applied to parse tree nodes. Given a node N on the parse tree of an SL sentence, the required reordering knowledge should enable the preprocessing module to determine how probable the children of N are reordered.[3] For simplicity, let us first consider the case of binary nodes only. Let N_1 and N_2, which yield phrases p_1 and p_2 respectively, be the child nodes of N. We want to determine the order of p_1 and p_2 with respect to their TL counterparts, $T(p_1)$ and $T(p_2)$. The knowledge for making such a decision can be learned from a word-aligned parallel corpus. There are two questions involved in obtaining training instances:

- How to define $T(p_i)$?
- How to define the order of $T(p_i)$s?

For the first question, we adopt a similar method as in (Fox, 2002): given an SL phrase $p_s = s_1 \dots s_i \dots s_n$ and a word alignment matrix A, we can enumerate the set of TL words $\{t_i : t_i \epsilon A(s_i)\}$, and then arrange the words in the order as they appear in the TL sentence. Let $\mathit{first}(t)$ be the first word in this sorted set and $\mathit{last}(t)$ be the last word. $T(p_s)$ is defined as the phrase $\mathit{first}(t) \dots \mathit{last}(t)$ in the TL sentence. Note that $T(p_s)$ may contain words not in the set $\{t_i\}$.

The question of the order of two TL phrases is not a trivial one. Since a word alignment matrix usually contains a lot of noises as well as one-to-many and many-to-many alignments, two TL phrases may overlap with each other. For the sake of the quality of reordering knowledge, if $T(p_1)$ and $T(p_2)$ overlap, then the node N with children N_1 and N_2 is not taken as a training instance. Obviously it will greatly reduce the amount of training input. To remedy data sparseness, less probable alignment points are removed so as to minimize overlapping phrases, since, after removing some alignment point, one of the TL phrases may become shorter and the two phrases may no longer overlap. The implementation is similar to the idea of *lexical weight* in (Koehn et al., 2003): all points in the alignment matrices of the entire training corpus are collected to calculate the probabilistic distribution, $P(t|s)$, of some TL word

[3] Some readers may prefer the expression *the subtree rooted at node N* to *node N*. The latter term is used in this paper for simplicity.

t given some SL word s. Any pair of overlapping $T(p_i)$s will be redefined by iteratively removing less probable word alignments until they no longer overlap. If they still overlap after all one/many-to-many alignments have been removed, then the refinement will stop and N, which covers p_is, is no longer taken as a training instance.

In sum, given a bilingual training corpus, a parser for the SL, and a word alignment tool, we can collect all binary parse tree nodes, each of which may be an instance of the required reordering knowledge. The next question is what kind of reordering knowledge can be formed out of these training instances. Two forms of reordering knowledge are investigated:

1. Reordering Rules, which have the form

$$Z:\ X\ Y \Rightarrow \begin{cases} X\ Y & Pr(\text{IN-ORDER}) \\ Y\ X & Pr(\text{INVERTED}) \end{cases}$$

   where Z is the phrase label of a binary node and X and Y are the phrase labels of Z's children, and $Pr(\text{INVERTED})$ and $Pr(\text{IN-ORDER})$ are the probability that X and Y are inverted on TL side and that not inverted, respectively. The probability figures are estimated by Maximum Likelihood Estimation.

2. Maximum Entropy (ME) Model, which does the binary classification whether a binary node's children are inverted or not, based on a set of features over the SL phrases corresponding to the two children nodes. The features that we investigated include the leftmost, rightmost, head, and context words[4], and their POSs, of the SL phrases, as well as the phrase labels of the SL phrases and their parent.

4 The Application of Reordering Knowledge

After learning reordering knowledge, the preprocessing module can apply it to the parse tree, t_S, of an SL sentence S and obtain the n-best list of S'. Since a ranking of S' is needed, we need some way to score each S'. Here probability is used as the scoring metric. In this section it is explained how the n-best reorderings of S and their associated scores/probabilites are computed.

Let us first look into the scoring of a particular reordering. Let $Pr(p \rightarrow p')$ be the probability of reordering a phrase p into p'. For a phrase q yielded by a non-binary node, there is only one 'reordering' of q, viz. q itself, thus $Pr(q \rightarrow q) = 1$. For a phrase p yielded by a binary node N, whose left child N_1 has reorderings p_1^i and right child N_2 has the reorderings p_2^j $(1 \leq i, j \leq n)$, p' has the form $p_1^i p_2^j$ or $p_2^j p_1^i$. Therefore, $Pr(p \rightarrow p') =$

$$\begin{cases} Pr(\text{IN-ORDER}) \times Pr(p_1^i \rightarrow p_1^{i'}) \times Pr(p_2^j \rightarrow p_2^{j'}) \\ Pr(\text{INVERTED}) \times Pr(p_2^j \rightarrow p_2^{j'}) \times Pr(p_1^i \rightarrow p_1^{i'}) \end{cases}$$

The figures $Pr(\text{IN-ORDER})$ and $Pr(\text{INVERTED})$ are obtained from the learned reordering knowledge. If reordering knowledge is represented as rules, then the required probability is the probability associated with the rule that can apply to N. If reordering knowledge is represented as an ME model, then the required probability is:

$$P(r|N) = \frac{\exp(\sum_i \lambda_i f_i(N, r))}{\sum_{r'} \exp(\sum_i \lambda_i f_i(N, r'))}$$

where $r \epsilon \{\text{IN-ORDER}, \text{INVERTED}\}$, and f_i's are features used in the ME model.

Let us turn to the computation of the n-best reordering list. Let $R(N)$ be the number of reorderings of the phrase yielded by N, then:

$$R(N) = \begin{cases} 2R(N_1)R(N_2) & \text{if } N \text{ has children } N_1, N_2 \\ 1 & \text{otherwise} \end{cases}$$

It is easily seen that the number of S's increases exponentially. Fortunately, what we need is merely an n-best list rather than a full list of reorderings. Starting from the leaves of t_S, for each node N covering phrase p, we only keep track of the n p's that have the highest reordering probability. Thus $R(N) \leq n$. There are at most $2n^2$ reorderings for any node and only the top-scored n reorderings are recorded. The n-best reorderings of S, i.e. the n-best reorderings of the yield of the root node of t_S, can be obtained by this efficient bottom-up method.

5 The Generalization of Reordering Knowledge

In the last two sections reordering knowledge is learned from and applied to binary parse tree nodes

[4]The context words of the SL phrases are the word to the left of the left phrase and the word to the right of the right phrase.

only. It is not difficult to generalize the theory of reordering knowledge to nodes of other branching factors. The case of binary nodes is simple as there are only two possible reorderings. The case of 3-ary nodes is a bit more complicated as there are six.[5] In general, an n-ary node has $n!$ possible reorderings of its children. The maximum entropy model has the same form as in the binary case, except that there are more classes of reordering patterns as n increases. The form of reordering rules, and the calculation of reordering probability for a particular node, can also be generalized easily.[6] The only problem for the generalized reordering knowledge is that, as there are more classes, data sparseness becomes more severe.

6 The Decoder

The last three sections explain how the $S \rightarrow n \times S'$ part of formula 2 is done. The $S' \rightarrow T$ part is simply done by our re-implementation of PHARAOH (Koehn, 2004). Note that non-monotonous translation is used here since the distance-based model is needed for local reordering. For the $n \times T \rightarrow \hat{T}$ part, the factors in consideration include the score of T returned by the decoder, and the reordering probability $Pr(S \rightarrow S')$. In order to conform to the log-linear model used in the decoder, we integrate the two factors by defining the total score of T as formula 3:

$$\exp(\lambda_r \log Pr(S \rightarrow S') + \sum_i \lambda_i F_i(S' \rightarrow T)) \quad (3)$$

The first term corresponds to the contribution of syntax-based reordering, while the second term that of the features F_i used in the decoder. All the feature weights (λs) were trained using our implementation of Minimum Error Rate Training (Och, 2003). The final translation $\hat{T}$ is the T with the highest total score.

[5] Namely, $N_1N_2N_3$, $N_1N_3N_2$, $N_2N_1N_3$, $N_2N_3N_1$, $N_3N_1N_2$, and $N_3N_2N_1$, if the child nodes in the original order are N_1, N_2, and N_3.

[6] For example, the reordering probability of a phrase $p = p_1p_2p_3$ generated by a 3-ary node N is

$$Pr(r) \times Pr(p_1^i) \times Pr(p_2^j) \times Pr(p_3^k)$$

where r is one of the six reordering patterns for 3-ary nodes.

It is observed in pilot experiments that, for a lot of long sentences containing several clauses, only one of the clauses is reordered. That is, our greedy reordering algorithm (c.f. section 4) has a tendency to focus only on a particular clause of a long sentence.

The problem was remedied by modifying our decoder such that it no longer translates a sentence at once; instead the new decoder does:

1. split an input sentence S into clauses $\{C_i\}$;
2. obtain the reorderings among $\{C_i\}$, $\{S_j\}$;
3. for each S_j, do
   (a) for each clause C_i in S_j, do
      i. reorder C_i into n-best C_i's,
      ii. translate each C_i' into $T(C_i')$,
      iii. select $\hat{T}(C_i')$;
   (b) concatenate $\{\hat{T}(C_i')\}$ into T_j;
4. select $\hat{T}_j$.

Step 1 is done by checking the parse tree if there are any IP or CP nodes[7] immediately under the root node. If yes, then all these IPs, CPs, and the remaining segments are treated as clauses. If no, then the entire input is treated as one single clause. Step 2 and step 3(a)(i) still follow the algorithm in section 4. Step 3(a)(ii) is trivial, but there is a subtle point about the calculation of language model score: the language model score of a translated clause is not independent from other clauses; it should take into account the last few words of the previous translated clause. The best translated clause $\hat{T}(C_i')$ is selected in step 3(a)(iii) by equation 3. In step 4 the best translation $\hat{T}_j$ is

$$\arg\max_{T_j} \exp(\lambda_r \log Pr(S \rightarrow S_j) + \sum_i score(T(C_i'))).$$

7 Experiments

7.1 Corpora

Our experiments are about Chinese-to-English translation. The NIST MT-2005 test data set is used for evaluation. (Case-sensitive) BLEU-4 (Papineni et al., 2002) is used as the evaluation metric. The

[7] IP stands for *inflectional phrase* and CP for *complementizer phrase*. These two types of phrases are *clauses* in terms of the Government and Binding Theory.

Branching Factor	2	3	>3
Count	12294	3173	1280
Percentage	73.41	18.95	7.64

Table 1: Distribution of Parse Tree Nodes with Different Branching Factors Note that nodes with only one child are excluded from the survey as reordering does not apply to such nodes.

test set and development set of NIST MT-2002 are merged to form our development set. The training data for both reordering knowledge and translation table is the one for NIST MT-2005. The GIGAWORD corpus is used for training language model. The Chinese side of all corpora are segmented into words by our implementation of (Gao et al., 2003).

7.2 The Preprocessing Module

As mentioned in section 3, the preprocessing module for reordering needs a parser of the SL, a word alignment tool, and a Maximum Entropy training tool. We use the Stanford parser (Klein and Manning, 2003) with its default Chinese grammar, the GIZA++ (Och and Ney, 2000) alignment package with its default settings, and the ME tool developed by (Zhang, 2004).

Section 5 mentions that our reordering model can apply to nodes of any branching factor. It is interesting to know how many branching factors should be included. The distribution of parse tree nodes as shown in table 1 is based on the result of parsing the Chinese side of NIST MT-2002 test set by the Stanford parser. It is easily seen that the majority of parse tree nodes are binary ones. Nodes with more than 3 children seem to be negligible. The 3-ary nodes occupy a certain proportion of the distribution, and their impact on translation performance will be shown in our experiments.

7.3 The decoder

The data needed by our Pharaoh-like decoder are translation table and language model. Our 5-gram language model is trained by the SRI language modeling toolkit (Stolcke, 2002). The translation table is obtained as described in (Koehn et al., 2003), i.e. the alignment tool GIZA++ is run over the training data in both translation directions, and the two alignment matrices are integrated by the GROW-DIAG-FINAL method into one matrix, from which phrase translation probabilities and lexical weights of both directions are obtained.

Test	Setting	BLEU
B1	standard phrase-based SMT	29.22
B2	(B1) + clause splitting	29.13

Table 2: Experiment Baseline

Test	Setting	BLEU 2-ary	BLEU 2,3-ary
1	rule	29.77	30.31
2	ME (phrase label)	29.93	30.49
3	ME (left,right)	30.10	30.53
4	ME ((3)+head)	30.24	30.71
5	ME ((3)+phrase label)	30.12	30.30
6	ME ((4)+context)	30.24	30.76

Table 3: Tests on Various Reordering Models
The 3rd column comprises the BLEU scores obtained by reordering binary nodes only, the 4th column the scores by reordering both binary and 3-ary nodes. The features used in the ME models are explained in section 3.

The most important system parameter is, of course, distortion limit. Pilot experiments using the standard phrase-based model show that the optimal distortion limit is 4, which was therefore selected for all our experiments.

7.4 Experiment Results and Analysis

The baseline of our experiments is the standard phrase-based model, which achieves, as shown by table 2, the BLEU score of 29.22. From the same table we can also see that the clause splitting mechanism introduced in section 6 does not significantly affect translation performance.

Two sets of experiments were run. The first set, of which the results are shown in table 3, tests the effect of different forms of reordering knowledge. In all these tests only the top 10 reorderings of each clause are generated. The contrast between tests 1 and 2 shows that ME modeling of reordering outperforms reordering rules. Tests 3 and 4 show that phrase labels can achieve as good performance as the lexical features of mere leftmost and rightmost words. However, when more lexical features

Input	海南省 2005年 还 将 继续 增加 对 公共 服务 和 社会 事业 基础 设施 投资
Reference	Hainan province will continue to increase its investment in the public services and social services infrastructures in 2005
Baseline	Hainan Province in 2005 will continue to increase for the public service and social infrastructure investment
Translation with Preprocessing	Hainan Province in 2005 will continue to increase investment in public services and social infrastructure

Table 4: Translation Example 1

Test	Setting	BLEU
a	length constraint	30.52
b	DL=0	30.48
c	n=100	30.78

Table 5: Tests on Various Constraints

are added (tests 4 and 6), phrase labels can no longer compete with lexical features. Surprisingly, test 5 shows that the combination of phrase labels and lexical features is even worse than using either phrase labels or lexical features only.

Apart from quantitative evaluation, let us consider the translation example of test 6 shown in table 4. To generate the correct translation, a phrase-based decoder should, after translating the word "增加" as "*increase*", jump to the last word "投资(investment)". This is obviously out of the capability of the baseline model, and our approach can accomplish the desired reordering as expected.

By and large, the experiment results show that no matter what kind of reordering knowledge is used, the preprocessing of syntax-based reordering does greatly improve translation performance, and that the reordering of 3-ary nodes is crucial.

The second set of experiments test the effect of some constraints. The basic setting is the same as that of test 6 in the first experiment set, and reordering is applied to both binary and 3-ary nodes. The results are shown in table 5.

In test (a), the constraint is that the module does not consider any reordering of a node if the yield of this node contains not more than four words. The underlying rationale is that reordering within distortion limit should be left to the distance-based model during decoding, and syntax-based reordering should focus on global reordering only. The result shows that this hypothesis does not hold. In practice syntax-based reordering also helps local reordering. Consider the translation example of test (a) shown in table 6. Both the baseline model and our model translate in the same way up to the word "围绕" (which is incorrectly translated as "*and*"). From this point, the proposed preprocessing model correctly jump to the last phrase "进行 了 讨论/*discussed*", while the baseline model fail to do so for the best translation. It should be noted, however, that there are only four words between "围绕" and the last phrase, and the desired order of decoding is within the capability of the baseline system. With the feature of syntax-based global reordering, a phrase-based decoder performs better even with respect to local reordering. It is because syntax-based reordering adds more weight to a hypothesis that moves words across longer distance, which is penalized by the distance-based model.

In test (b) distortion limit is set as 0; i.e. reordering is done merely by syntax-based preprocessing. The worse result is not surprising since, after all, preprocessing discards many possibilities and thus reduce the search space of the decoder. Some local reordering model is still needed during decoding.

Finally, test (c) shows that translation performance does not improve significantly by raising the number of reorderings. This implies that our approach is very efficient in that only a small value of n is capable of capturing the most important global reordering patterns.

8 Conclusion and Future Work

This paper proposes a novel, probabilistic approach to reordering which combines the merits of syntax and phrase-based SMT. On the one hand, global reordering, which cannot be accomplished by the

Input	与此同时 , 尤先科 和 助手 围绕 组建 新 政府 问题 进行 了 讨论
Reference	Meanwhile , Yushchenko and his assistants discussed issues concerning the establishment of a new government
Baseline	The same time , Yushchenko assistants and a new Government on issues discussed
Translation with Preprocessing	The same time , Yushchenko assistants and held discussions on the issue of a new government

Table 6: Translation Example 2

phrase-based model, is enabled by the tree operations in preprocessing. On the other hand, local reordering is preserved and even strengthened in our approach. Experiments show that, for the NIST MT-05 task of Chinese-to-English translation, the proposal leads to BLEU improvement of 1.56%.

Despite the encouraging experiment results, it is still not very clear how the syntax-based and distance-based models complement each other in improving word reordering. In future we need to investigate their interaction and identify the contribution of each component. Moreover, it is observed that the parse trees returned by a full parser like the Stanford parser contain too many nodes which seem not be involved in desired reorderings. Shallow parsers should be tried to see if they improve the quality of reordering knowledge.

References

Yaser Al-Onaizan, and Kishore Papineni. 2006. Distortion Models for Statistical Machine Translation. *Proceedings for ACL 2006*.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause Restructuring for Statistical Machine Translation. *Proceedings for ACL 2005*.

M.R. Costa-jussà, and J.A.R. Fonollosa. 2006. Statistical Machine Reordering. *Proceedings for EMNLP 2006*.

Heidi Fox. 2002. Phrase Cohesion and Statistical Machine Translation. *Proceedings for EMNLP 2002*.

Jianfeng Gao, Mu Li, and Chang-Ning Huang 2003. Improved Source-Channel Models for Chinese Word Segmentation. *Proceedings for ACL 2003*.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. *Proceedings for ACL 2003*.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical Phrase-based Translation. *Proceedings for HLT-NAACL 2003*.

Philipp Koehn. 2004. Pharaoh: a Beam Search Decoder for Phrase-Based Statistical Machine Translation Models. *Proceedings for AMTA 2004*.

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. *Proceedings for IWSLT 2005*.

Franz J. Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. *Proceedings for ACL 2003*.

Franz J. Och, and Hermann Ney. 2000. Improved Statistical Alignment Models. *Proceedings for ACL 2000*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. *Proceedings for ACL 2002*.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency Treelet Translation: Syntactically Informed Phrasal SMT. *Proceedings for ACL 2005*.

Andreas Stolcke. 2002. SRILM - An Extensible Language Modeling Toolkit. *Proceedings for the International Conference on Spoken Language Understanding 2002*.

Christoph Tillmann. 2004. A Unigram Orientation Model for Statistical Machine Translation. *Proceedings for ACL 2004*.

Fei Xia, and Michael McCord 2004. Improving a Statistical MT System with Automatically Learned Rewrite Patterns. *Proceedings for COLING 2004*.

Kenji Yamada, and Kevin Knight. 2001. A syntax-based statistical translation model. *Proceedings for ACL 2001*.

Le Zhang. 2004. Maximum Entropy Modeling Toolkit for Python and C++. http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html.

Machine Translation by Triangulation: Making Effective Use of Multi-Parallel Corpora

Trevor Cohn and **Mirella Lapata**
Human Computer Research Centre, School of Informatics
University of Edinburgh
{tcohn,mlap}@inf.ed.ac.uk

Abstract

Current phrase-based SMT systems perform poorly when using small training sets. This is a consequence of unreliable translation estimates and low coverage over source and target phrases. This paper presents a method which alleviates this problem by exploiting multiple translations of the same source phrase. Central to our approach is *triangulation*, the process of translating from a source to a target language via an intermediate third language. This allows the use of a much wider range of parallel corpora for training, and can be combined with a standard phrase-table using conventional smoothing methods. Experimental results demonstrate BLEU improvements for triangulated models over a standard phrase-based system.

1 Introduction

Statistical machine translation (Brown et al., 1993) has seen many improvements in recent years, most notably the transition from word- to phrase-based models (Koehn et al., 2003). Modern SMT systems are capable of producing high quality translations when provided with large quantities of training data. With only a small training sample, the translation output is often inferior to the output from using larger corpora because the translation algorithm must rely on more sparse estimates of phrase frequencies and must also 'back-off' to smaller sized phrases. This often leads to poor choices of target phrases and reduces the coherence of the output. Unfortunately, parallel corpora are not readily available in large quantities, except for a small subset of the world's languages (see Resnik and Smith (2003) for discussion), therefore limiting the potential use of current SMT systems.

In this paper we provide a means for obtaining more reliable translation frequency estimates from small datasets. We make use of *multi-parallel* corpora (sentence aligned parallel texts over three or more languages). Such corpora are often created by international organisations, the United Nations (UN) being a prime example. They present a challenge for current SMT systems due to their relatively moderate size and domain variability (examples of UN texts include policy documents, proceedings of meetings, letters, *etc.*). Our method translates each target phrase, $\mathbf{t}$, first to an intermediate language, $\mathbf{i}$, and then into the source language, $\mathbf{s}$. We call this two-stage translation process *triangulation* (Kay, 1997). We present a probabilistic formulation through which we can estimate the desired phrase translation distribution (phrase-table) by marginalisation, $p(\mathbf{s}|\mathbf{t}) = \sum_{\mathbf{i}} p(\mathbf{s}, \mathbf{i}|\mathbf{t})$.

As with conventional smoothing methods (Koehn et al., 2003; Foster et al., 2006), triangulation increases the robustness of phrase translation estimates. In contrast to smoothing, our method alleviates data sparseness by exploring additional multi-parallel data rather than adjusting the probabilities of existing data. Importantly, triangulation provides us with separately estimated phrase-tables which could be further smoothed to provide more reliable distributions. Moreover, the triangulated phrase-tables can be easily combined with the standard source-target phrase-table, thereby improving the coverage over unseen source phrases.

As an example, consider Figure 1 which shows the coverage of unigrams and larger n-gram phrases when using a standard source target phrase-table, a triangulated phrase-table with one (*it*) and nine languages (*all*), and a combination of standard and triangulated phrase-tables (*all+standard*). The phrases were harvested from a small French-English bitext

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 728–735,
Prague, Czech Republic, June 2007.

and evaluated against a test set. Although very few small phrases are unknown, the majority of larger phrases are unseen. The *Italian* and *all* results show that triangulation alone can provide similar or improved coverage compared to the standard source-target model; further improvement is achieved by combining the triangulated and standard models (*all+standard*). These models and datasets will be described in detail in Section 3.

We also demonstrate that triangulation can be used on its own, that is *without a source-target distribution*, and still yield acceptable translation output. This is particularly heartening, as it provides a means of translating between the many "low density" language pairs for which we don't yet have a source-target bitext. This allows SMT to be applied to a much larger set of language pairs than was previously possible.

In the following section we provide an overview of related work. Section 3 introduces a generative formulation of triangulation. We present our evaluation framework in Section 4 and results in Section 5.

2 Related Work

The idea of using multiple source languages for improving the translation quality of the target language dates back at least to Kay (1997), who observed that ambiguities in translating from one language onto another may be resolved if a translation into some third language is available. Systems which have used this notion of triangulation typically create several candidate sentential target translations for source sentences via different languages. A single translation is then selected by finding the candidate that yields the best overall score (Och and Ney, 2001; Utiyama and Isahara, 2007) or by co-training (Callison-Burch and Osborne, 2003). This ties in with recent work on ensemble combinations of SMT systems, which have used alignment techniques (Matusov et al., 2006) or simple heuristics (Eisele, 2005) to guide target sentence selection and generation. Beyond SMT, the use of an intermediate language as a translation aid has also found application in cross-lingual information retrieval (Gollins and Sanderson, 2001).

Callison-Burch et al. (2006) propose the use of paraphrases as a means of dealing with unseen source phrases. Their method acquires paraphrases by identifying candidate phrases in the source language, translating them into multiple target languages, and then back to the source. Unknown source phrases are substituted by the back-translated paraphrases and translation proceeds on the paraphrases.

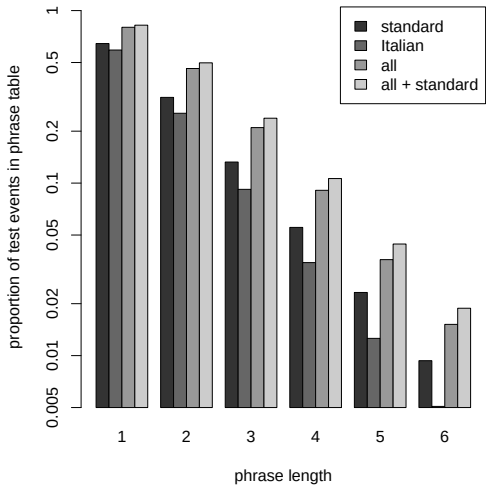


Figure 1: Coverage of *fr* → *en* test phrases using a 10,000 sentence bitext. The standard model is shown alongside triangulated models using one (Italian) or nine other languages (all).

In line with previous work, we exploit multiple source corpora to alleviate data sparseness and increase translation coverage. However, we differ in several important respects. Our method operates over phrases rather than sentences. We propose a generative formulation which treats triangulation not as a post-processing step but as part of the translation model itself. The induced phrase-table entries are fed directly into the decoder, thus avoiding the additional inefficiencies of merging the output of several translation systems.

Although related to Callison-Burch et al. (2006) our method is conceptually simpler and more general. Phrase-table entries are created via multiple source languages without the intermediate step of paraphrase extraction, thereby reducing the exposure to compounding errors. Our phrase-tables may well contain paraphrases but these are naturally induced as part of our model, without extra processing effort. Furthermore, we improve the translation estimates for both seen and unseen phrase-table entries, whereas Callison-Burch et al. concentrate solely on unknown phrases. In contrast to Utiyama and Isahara (2007), we employ a large number of intermediate languages and demonstrate how triangulated phrase-tables can be combined with standard phrase-tables to improve translation output.

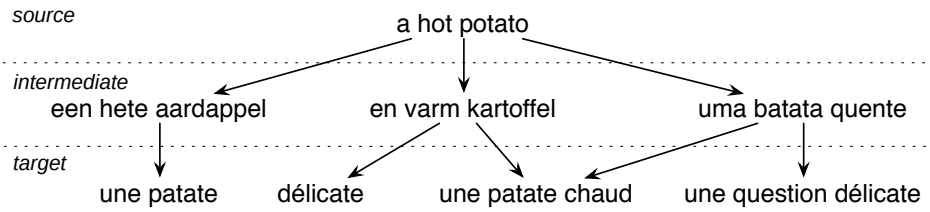


Figure 2: Triangulation between English (source) and French (target), showing three phrases in Dutch, Danish and Portuguese, respectively. Arrows denote phrases aligned in a language pair and also the generative translation process.

3 Triangulation

We start with a motivating example before formalising the mechanics of triangulation. Consider translating the English phrase *a hot potato*[1] into French, as shown in Figure 2. In our corpus this English phrase occurs only three times. Due to errors in the word alignment the phrase was not included in the English-French phrase-table. Triangulation first translates *hot potato* into a set of intermediate languages (Dutch, Danish and Portuguese are shown in the figure), and then these phrases are further translated into the target language (French). In the example, four different target phrases are obtained, all of which are useful phrase-table entries. We argue that the redundancy introduced by a large suite of other languages can correct for errors in the word alignments and also provide greater generalisation, since the translation distribution is estimated from a richer set of data-points. For example, instances of the Danish *en varm kartoffel* may be used to translate several English phrases, not only *a hot potato*.

In general we expect that a wider range of possible translations are found for any source phrase, simply due to the extra layer of indirection. So, if a source phrase tends to align with two different target phrases, then we would also expect it to align with two phrases in the 'intermediate' language. These intermediate phrases should then each align with two target phrases, yielding up to four target phrases. Consequently, triangulation will often produce more varied translation distributions than the standard source-target approach.

3.1 Formalisation

We now formalise triangulation as a generative probabilistic process operating independently on phrase pairs. We start with the conditional distribution over three languages, $p(\mathbf{s}, \mathbf{i}|\mathbf{t})$, where the arguments denote phrases in the source, intermediate and target language, respectively. From this distribution, we can find the desired conditional over the source-target pair by marginalising out the intermediate phrases:[2]

$$p(\mathbf{s}|\mathbf{t}) = \sum_{\mathbf{i}} p(\mathbf{s}|\mathbf{i}, \mathbf{t}) p(\mathbf{i}|\mathbf{t}) \approx \sum_{\mathbf{i}} p(\mathbf{s}|\mathbf{i}) p(\mathbf{i}|\mathbf{t}) \quad (1)$$

where (1) imposes a simplifying conditional independence assumption: the intermediate phrase fully represents the information (semantics, syntax, *etc.*) in the source phrase, rendering the target phrase redundant in $p(\mathbf{s}|\mathbf{i}, \mathbf{t})$.

Equation (1) requires that all phrases in the intermediate-target bitext must also be found in the source-intermediate bitext, such that $p(\mathbf{s}|\mathbf{i})$ is defined. Clearly this will often not be the case. In these situations we could back-off to another distribution (by discarding part, or all, of the conditioning context), however we take a more pragmatic approach and ignore the missing phrases. This problem of missing contexts is uncommon in multi-parallel corpora, but is more common when the two bitexts are drawn from different sources.

While triangulation is intuitively appealing, it may suffer from a few problems. Firstly, as with any SMT approach, the translation estimates are based on noisy automatic word alignments. This leads to many errors and omissions in the phrase-table. With a standard source-target phrase-table these errors are only encountered once, however with triangulation they are encountered twice, and therefore the errors will compound. This leads to more noisy estimates than in the source-target phrase-table.

Secondly, the increased exposure to noise means that triangulation will omit a greater proportion of large or rare phrases than the standard method. An

[1] An idiom meaning a situation for which no one wants to claim responsibility.

[2] Equation (1) is used with the source and target arguments reversed to give $p(\mathbf{t}|\mathbf{s})$.

alignment error in either of the source-intermediate or intermediate-target bitexts can prevent the extraction of a source-target phrase pair. This effect can be seen in Figure 1, where the coverage of the Italian triangulated phrase-table is worse than the standard source-target model, despite the two models using the same sized bitexts. As we explain in the next section, these problems can be ameliorated by using the triangulated phrase-table in conjunction with a standard phrase-table.

Finally, another potential problem stems from the independence assumption in (1), which may be an oversimplification and lead to a loss of information. The experiments in Section 5 show that this effect is only mild.

3.2 Merging the phrase-tables

Once induced, the triangulated phrase-table can be usefully combined with the standard source-target phrase-table. The simplest approach is to use linear interpolation to combine the two (or more) distributions, as follows:

$$p(\mathbf{s}, \mathbf{t}) = \sum_j \lambda_j p_j(\mathbf{s}, \mathbf{t}) \quad (2)$$

where each joint distribution, p_j, has a non-negative weight, λ_j, and the weights sum to one. The joint distribution for triangulated phrase-tables is defined in an analogous way to Equation (1). We expect that the standard phrase-table should be allocated a higher weight than triangulated phrase-tables, as it will be less noisy. The joint distribution is now conditionalised to yield $p(\mathbf{s}|\mathbf{t})$ and $p(\mathbf{t}|\mathbf{s})$, which are both used as features in the decoder. Note that the resulting conditional distribution will be drawn solely from one input distribution when the conditioning context is unseen in the remaining distributions. This may lead to an over-reliance on unreliable distributions, which can be ameliorated by smoothing (e.g., Foster et al. (2006)).

As an alternative to linear interpolation, we also employ a weighted product for phrase-table combination:

$$p(\mathbf{s}|\mathbf{t}) \propto \prod_j p_j(\mathbf{s}|\mathbf{t})^{\lambda_j} \quad (3)$$

This has the same form used for log-linear training of SMT decoders (Och, 2003), which allows us to treat each distribution as a feature, and learn the mixing weights automatically. Note that we must individually smooth the component distributions in (3) to stop zeros from propagating. For this we use Simple Good-Turing smoothing (Gale and Sampson, 1995) for each distribution, which provides estimates for zero count events.

4 Experimental Design

Corpora We used the Europarl corpus (Koehn, 2005) for experimentation. This corpus consists of about 700,000 sentences of parliamentary proceedings from the European Union in eleven European languages. We present results on the full corpus for a range of language pairs. In addition, we have created smaller parallel corpora by sub-sampling 10,000 sentence bitexts for each language pair. These corpora are likely to have minimal overlap — about 1.5% of the sentences will be shared between each pair. However, the phrasal overlap is much greater (10 to 20%), which allows for triangulation using these common phrases. This training setting was chosen to simulate translating to or from a "low density" language, where only a few small independently sourced parallel corpora are available. These bitexts were used for direct translation and triangulation. All experimental results were evaluated on the ACL/WMT 2005[3] set of 2,000 sentences, and are reported in BLEU percentage-points.

Decoding Pharaoh (Koehn, 2003), a beam-search decoder, was used to maximise:

$$\mathbf{T}^* = \arg\max_{\mathbf{T}} \prod_j f_j(\mathbf{T}, \mathbf{S})^{\lambda_j} \quad (4)$$

where $\mathbf{T}$ and $\mathbf{S}$ denote a target and source sentence respectively. The parameters, λ_j, were trained using minimum error rate training (Och, 2003) to maximise the BLEU score (Papineni et al., 2002) on a 150 sentence development set. We used a standard set of features, comprising a 4-gram language model, distance based distortion model, forward and backward translation probabilities, forward and backward lexical translation scores and the phrase- and word-counts. The translation models and lexical scores were estimated on the training corpus which was automatically aligned using Giza++ (Och et al., 1999) in both directions between source and target and symmetrised using the growing heuristic (Koehn et al., 2003).

[3]For details see `http://www.statmt.org/wpt05/mt-shared-task`.

Lexical weights The lexical translation score is used for smoothing the phrase-table translation estimate. This represents the translation probability of a phrase when it is decomposed into a series of independent word-for-word translation steps (Koehn et al., 2003), and has proven a very effective feature (Zens and Ney, 2004; Foster et al., 2006). Pharaoh's lexical weights require access to word-alignments; calculating these alignments between the source and target words in a phrase would prove difficult for a triangulated model. Therefore we use a modified lexical score, corresponding to the maximum IBM model 1 score for the phrase pair:

$$lex(\mathbf{t}|\mathbf{s}) = \frac{1}{Z} \max_{\mathbf{a}} \prod_{k} p(t_k|s_{a_k}) \qquad (5)$$

where the maximisation[4] ranges over all one-to-many alignments and Z normalises the score by the number of possible alignments.

The lexical probability is obtained by interpolating a relative frequency estimate on the source-target bitext with estimates from triangulation, in the same manner used for phrase translations in (1) and (2). The addition of the lexical probability feature yielded a substantial gain of up to two BLEU points over a basic feature set.

5 Experimental Results

The evaluation of our method was motivated by three questions: (1) How do different training requirements affect the performance of the triangulated models presented in this paper? We expect performance gains with triangulation on small and moderate datasets. (2) Is machine translation output influenced by the choice of the intermediate language/s? Here, we would like to evaluate whether the number and choice of intermediate languages matters. (3) What is the quality of the triangulated phrase-table? In particular, we are interested in the resulting distribution and whether it is sufficiently distinct from the standard phrase-table.

5.1 Training requirements

Before reporting our results, we briefly discuss the specific choice of model for our experiments. As mentioned in Section 3, our method combines the triangulated phrase-table with the standard source-target one. This is desired in order to compensate for the noise incurred by the triangulation process. We used two combination methods, namely linear interpolation (see (2)) and a weighted geometric mean (see (3)).

	standard	interp	+indic	separate
en → *de*	12.03	12.66	12.95	12.25
fr → *en*	23.02	24.63	23.86	23.43

Table 1: Different feature sets used with the 10K training corpora, using a single language (*es*) for triangulation. The columns refer to standard, uniform interpolation, interpolation with 0-1 indicator features, and separate phrase-tables, respectively.

Table 1 reports the results for two translation tasks when triangulating with a single language (*es*) using three different feature sets, each with different translation features. The *interpolation* model uses uniform linear interpolation to merge the standard and triangulated phrase-tables. Non-uniform mixtures did not provide consistent gains, although, as expected, biasing towards the standard phrase-table was more effective than against. The *indicator* model uses the same interpolated distribution along with a series of 0-1 indicator features to identify the source of each event, *i.e.*, if each $(\mathbf{s}, \mathbf{t})$ pair is present in phrase-table j. We also tried per-context features with similar results. The *separate* model has a separate feature for each phrase-table.

All three feature sets improve over the standard source-target system, while the interpolated features provided the best overall performance. The relatively poorer performance of the separate model is perhaps surprising, as it is able to differentially weight the component distributions; this is probably due to MERT not properly handling the larger feature sets. In all subsequent experiments we report results using linear interpolation.

As a proof of concept, we first assessed the effect of triangulation on corpora consisting of 10,000 sentence bitexts. We expect triangulation to deliver performance gains on small corpora, since a large number of phrase-table entries will be unseen. In Table 2 each entry shows the BLEU score when using the standard phrase-table and the absolute improvement when using triangulation. Here we have used three languages for triangulation ($it \cup \{de, en, es, fr\} \setminus \{s, t\}$). The source-target languages were chosen so as to mirror the evaluation setup of NAACL/WMT. The translation tasks range

[4]The maximisation in (5) can be replaced with a sum with similar experimental results.

s ↓ t →	*de*	*en*	*es*	*fr*
de	-	17.58	16.84	18.06
	-	+1.20	+1.99	+1.94
en	12.45	-	23.83	24.05
	+1.22	-	+1.04	+1.48
es	12.31	23.83	-	32.69
	+2.24	+1.35	-	+0.85
fr	11.76	23.02	31.22	-
	+2.41	+2.24	+1.30	-

Table 2: BLEU improvements over the standard phrase-table (top) when interpolating with three triangulated phrase-tables (bottom) on the small training sample.

from easy (*es* → *fr*) to very hard (*de* → *en*). In all cases triangulation resulted in an improvement in translation quality, with the highest gains observed for the most difficult tasks (to and from German). For these tasks the standard systems have poor coverage (due in part to the sizeable vocabulary of German phrases) and therefore the gain can be largely explained by the additional coverage afforded by the triangulated phrase-tables.

To test whether triangulation can also improve performance of larger corpora we ran six separate translation tasks on the full Europarl corpus. The results are presented in Table 3, for a single triangulation language used alone (*triang*) or uniformly interpolated with the standard phrase-table (*interp*). These results show that triangulation can produce high quality translations on its own, which is noteworthy, as it allows for SMT between a much larger set of language pairs. Using triangulation in conjunction with the standard phrase-table improved over the standard system in most instances, and only degraded performance once. The improvement is largest for the German tasks which can be explained by triangulation providing better robustness to noisy alignments (which are often quite poor for German) and better estimates of low-count events. The difficulty of aligning German with the other languages is apparent from the Giza++ perplexity: the final Model 4 perplexities for German are quite high, as much as double the perplexity for more easily aligned language pairs (*e.g.*, Spanish-French).

Figure 3 shows the effect of triangulation on different sized corpora for the language pair *fr* → *en*. It presents learning curves for the standard system and a triangulated system using one language (*es*). As can be seen, gains from triangulation only diminish slightly for larger training corpora, and that

task	standard	interm	triang	interp
de → *en*	23.85	*es*	23.48	**24.36**
en → *de*	17.24	*es*	16.28	**17.42**
es → *en*	30.48	*fr*	29.06	**30.52**
en → *es*	**29.09**	*fr*	28.19	**29.09**
fr → *en*	29.66	*es*	29.59	**30.36**
en → *fr*	**30.07**	*es*	28.94	29.62

Table 3: Results on the full training set showing triangulation with a single language, both alone (triang) and alongside a standard model (interp).

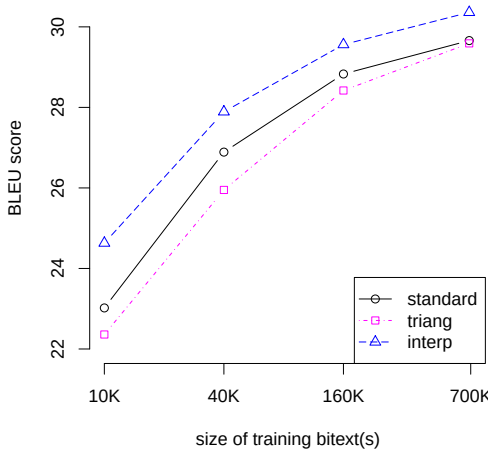


Figure 3: Learning curve for *fr* → *en* translation for the standard source-target model and a triangulated model using Spanish as an intermediate language.

the purely triangulated models have very competitive performance. The gain from interpolation with a triangulated model is roughly equivalent to having twice as much training data.

Finally, notice that triangulation may benefit when the sentences in each bitext are drawn from the same source, in that there are no unseen 'intermediate' phrases, and therefore (1) can be easily evaluated. We investigate this by examining the robustness of our method in the face of disjoint bitexts. The concepts contained in each bitext will be more varied, potentially leading to better coverage of the target language. In lieu of a study on different domain bitexts which we plan for the future, we bisected the Europarl corpus for *fr* → *en*, triangulating with Spanish. The triangulated models were presented with *fr-es* and *es-en* bitexts drawn from either the same half of the corpus or from different halves, resulting in scores of 28.37 and 28.13, respectively.[5] These results indicate that triangulation is effective

[5]The baseline source-target system on one half has a score of 28.85.

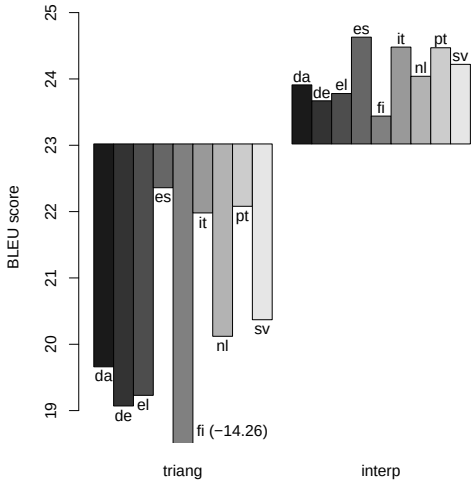


Figure 4: Comparison of different triangulation languages for *fr* → *en* translation, relative to the standard model (10K training sample). The bar for *fi* has been truncated to fit on the graph.

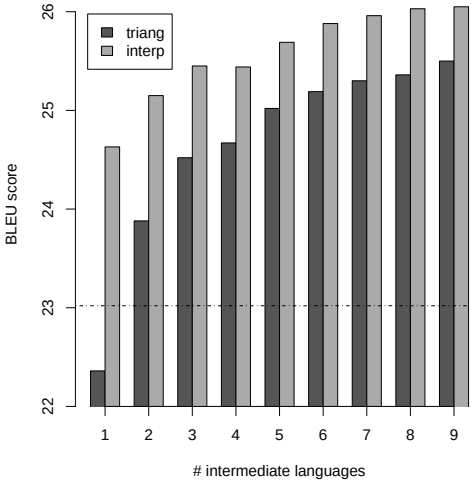


Figure 5: Increasing the number of intermediate languages used for triangulation increases performance for *fr* → *en* (10K training sample). The dashed line shows the BLEU score for the standard phrase-table.

for disjoint bitexts, although ideally we would test this with independently sourced parallel texts.

5.2 The choice of intermediate languages

The previous experiments used an ad-hoc choice of 'intermediate' language/s for triangulation, and we now examine which languages are most effective. Figure 4 shows the efficacy of the remaining nine languages when translating *fr* → *en*. Minimum error-rate training was not used for this experiment, or the next shown in Figure 5, in order to highlight the effect of the changing translation estimates. Romance languages (*es, it, pt*) give the best results, both on their own and when used together with the standard phrase-table (using uniform interpolation); Germanic languages (*de, nl, da, sv*) are a distant second, with the less related Greek and Finnish the least useful. Interpolation yields an improvement for all 'intermediate' languages, even Finnish, which has a very low score when used alone.

The same experiment was repeated for *en* → *de* translation with similar trends, except that the Germanic languages out-scored the Romance languages. These findings suggest that 'intermediate' languages which exhibit a high degree of similarity with the source or target language are desirable. We conjecture that this is a consequence of better automatic word alignments and a generally easier translation task, as well as a better preservation of information between aligned phrases.

Using a single language for triangulation clearly improves performance, but can we realise further improvements by using additional languages? Figure 5 shows the performance profile for *fr* → *en* when adding languages in a fixed order. The languages were ordered by family, with Romance before Germanic before Greek and Finnish. Each addition results in an increase in performance, even for the final languages, from which we expect little information. The purely triangulated (triang) and interpolated scores (interp) are converging, suggesting that the source-target bitext is redundant given sufficient triangulated data. We obtained similar results for *en* → *de*.

5.3 Evaluating the quality of the phrase-table

Our experimental results so far have shown that triangulation is not a mere approximation of the source-target phrase-table, but that it extracts additional useful translation information. We now assess the phrase-table quality more directly. Comparative statistics of a standard and a triangulated phrase-table are given in Table 4. The coverage over source and target phrases is much higher in the standard table than the triangulated tables, which reflects the reduced ability of triangulation to extract large phrases — despite the large increase in the number of events. The table also shows the overlapping probability mass which measures the sum of probability in one table for which the events are present in the other. This shows that the majority of mass is shared by both tables (as joint distributions), although there are significant differences. The Jensen-Shannon divergence is perhaps more appropriate for the comparison, giving a relatively high divergence

	standard	triang
source phrases (M)	8	2.5
target phrases (M)	7	2.5
events (M)	12	70
overlapping mass	0.646	0.750

Table 4: Comparative statistics of the standard triangulated table on *fr* → *en* using the full training set and Spanish as an intermediate language.

of 0.3937. This augurs well for the combination of standard and triangulated phrase-tables, where diversity is valued. The decoding results (shown in Table 3 for $fr \rightarrow en$) indicate that the two methods have similar efficacy, and that their interpolated combination provides the best overall performance.

6 Conclusion

In this paper we have presented a novel method for obtaining more reliable translation estimates from small datasets. The key premise of our work is that multi-parallel data can be usefully exploited for improving the coverage and quality of phrase-based SMT. Our triangulation method translates from a source to a target via one or many intermediate languages. We present a generative formulation of this process and show how it can be used together with the entries of a standard source-target phrase-table.

We observe large performance gains when translating with triangulated models trained on small datasets. Furthermore, when combined with a standard phrase-table, our models also yield performance improvements on larger datasets. Our experiments revealed that triangulation benefits from a large set of intermediate languages and that performance is increased when languages of the same family to the source or target are used as intermediates.

We have just scratched the surface of the possibilities for the framework discussed here. Important future directions lie in combining triangulation with richer means of conventional smoothing and using triangulation to translate between low-density language pairs.

Acknowledgements The authors acknowledge the support of EPSRC (grants GR/T04540/01 and GR/T04557/01). Special thanks to Markus Becker, Chris Callison-Burch, David Talbot and Miles Osborne for their helpful comments.

References

P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

C. Callison-Burch, M. Osborne. 2003. Bootstrapping parallel corpora. In *Proceedings of the NAACL Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, Edmonton, Canada.

C. Callison-Burch, P. Koehn, M. Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proceedings of the HLT/NAACL*, 17–24, New York, NY.

A. Eisele. 2005. First steps towards multi-engine machine translation. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, 155–158, Ann Arbor, MI.

G. Foster, R. Kuhn, H. Johnson. 2006. Phrase-table smoothing for statistical machine translation. In *Proceedings of the EMNLP*, 53–61, Sydney, Australia.

W. A. Gale, G. Sampson. 1995. Good-turing frequency estimation without tears. *Journal of Quantitative Linguistics*, 2(3):217–237.

T. Gollins, M. Sanderson. 2001. Improving cross language retrieval with triangulated translation. In *Proceedings of the SIGIR*, 90–95, New Orleans, LA.

M. Kay. 1997. The proper place of men and machines in language translation. *Machine Translation*, 12(1–2):3–23.

P. Koehn, F. J. Och, D. Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the HLT/NAACL*, 48–54, Edomonton, Canada.

P. Koehn. 2003. *Noun Phrase Translation*. Ph.D. thesis, University of Southern California, Los Angeles, California.

P. Koehn. 2005. Europarl: A parallel corpus for evaluation of machine translation. In *Proceedings of MT Summit*, Phuket, Thailand.

E. Matusov, N. Ueffing, H. Ney. 2006. Computing consesus translation from multiple machine translation systems using enhanced hypotheses alignment. In *Proceedings of the EACL*, 33–40, Trento, Italy.

F. J. Och, H. Ney. 2001. Statistical multi-source translation. In *Proceedings of the MT Summit*, 253–258, Santiago de Compostela, Spain.

F. J. Och, C. Tillmann, H. Ney. 1999. Improved alignment models for statistical machine translation. In *Proceedings of the EMNLP and VLC*, 20–28, University of Maryland, College Park, MD.

F. J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the ACL*, 160–167, Sapporo, Japan.

K. Papineni, S. Roukos, T. Ward, W.-J. Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the ACL*, 311–318, Philadelphia, PA.

P. Resnik, N. A. Smith. 2003. The Web as a parallel corpus. *Computational Linguistics*, 29(3):349–380.

M. Utiyama, H. Isahara. 2007. A comparison of pivot methods for phrase-based statistical machine translation. In *Proceedings of the HLT/NAACL*, 484–491, Rochester, NY.

R. Zens, H. Ney. 2004. Improvements in phrase-based statistical machine translation. In D. M. Susan Dumais, S. Roukos, eds., *Proceedings of the HLT/NAACL*, 257–264, Boston, MA.

A Maximum Expected Utility Framework for Binary Sequence Labeling

Martin Jansche*
jansche@acm.org

Abstract

We consider the problem of predictive inference for probabilistic binary sequence labeling models under F-score as utility. For a simple class of models, we show that the number of hypotheses whose expected F-score needs to be evaluated is linear in the sequence length and present a framework for efficiently evaluating the expectation of many common loss/utility functions, including the F-score. This framework includes both exact and faster inexact calculation methods.

1 Introduction

1.1 Motivation and Scope

The weighted F-score (van Rijsbergen, 1974) plays an important role in the evaluation of binary classifiers, as it neatly summarizes a classifier's ability to identify the positive class. A variety of methods exists for training classifiers that optimize the F-score, or some similar trade-off between false positives and false negatives, precision and recall, sensitivity and specificity, type I error and type II error rate, etc. Among the most general methods are those of Mozer et al. (2001), whose constrained optimization technique is similar to those in (Gao et al., 2006; Jansche, 2005). More specialized methods also exist, for example for support vector machines (Musicant et al., 2003) and for conditional random fields (Gross et al., 2007; Suzuki et al., 2006).

All of these methods are about classifier training. In this paper we focus primarily on the related, but orthogonal, issue of predictive inference with a fully trained probabilistic classifier. Using the weighted F-score as our utility function, predictive inference amounts to choosing an optimal hypothesis which maximizes the expected utility. We refer to this as the prediction or decoding task. In general, decoding can be a hard computational problem (Casacuberta and de la Higuera, 2000; Knight, 1999). In this paper we show that the maximum expected F-score decoding problem can be solved in polynomial time under certain assumptions about the underlying probability model. One key ingredient in our solution is a very general framework for evaluating the expected F-score, and indeed many other utility functions, of a fixed hypothesis.[1] This framework can also be applied to discriminative classifier training.

1.2 Background and Notation

We formulate our approach in terms of sequence labeling, although it has applications beyond that. This is motivated by the fact that our framework for evaluating expected utility is indeed applicable to general sequence labeling tasks, while our decoding method is more restricted. Another reason is that the F-score is only meaningful for comparing two (multi)sets or two binary sequences, but the notation for multisets is slightly more awkward.

All tasks considered here involve strings of binary labels. We write the length of a given string $y \in \{0,1\}^n$ as $|y| = n$. It is convenient to view such strings as real vectors – whose components happen to be 0 or 1 – with the dot product defined as usual. Then $y \cdot y$ is the number of ones that occur in the string y. For two strings x, y of the same length $|x| = |y|$ the number of ones that occur at corresponding indices is $x \cdot y$.

Given a hypothesis z and a gold standard label sequence y, we define the following quantities:

1. $T = y \cdot y$, the genuine positives;
2. $P = z \cdot z$, the predicted positives;
3. $A = z \cdot y$, the true positives (predicted positives that are genuinely positive);

*Current affiliation: Google Inc. Former affiliation: Center of Computational Learning Systems, Columbia University.

[1] A proof-of-concept implementation is available at `http://purl.org/net/jansche/meu_framework/`.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 736–743,
Prague, Czech Republic, June 2007.

4. $Recl = A/T$, recall (a.k.a. sensitivity or power);
5. $Prec = A/P$, precision.

The β-weighted F-score is then defined as the weighted harmonic mean of recall and precision. This simplifies to

$$F_\beta = \frac{(\beta+1)A}{P+\beta T} \qquad (\beta > 0) \tag{1}$$

where we assume for convenience that $0/0 \stackrel{\text{def}}{=} 1$ to avoid explicitly dealing with the special case of the denominator being zero. We will write the weighted F-score from now on as $F(z,y)$ to emphasize that it is a function of z and y.

1.3 Expected *F*-Score

In Section 3 we will develop a method for evaluating the expectation of the F-score, which can also be used as a smooth approximation of the raw F-score during classifier training: in that task (which we will not discuss further in this paper), z are the supervised labels, y is the classifier output, and the challenge is that $F(z,y)$ does not depend smoothly on the parameters of the classifier. Gradient-based optimization techniques are not applicable unless some of the quantities defined above are replaced by approximations that depend smoothly on the classifier's parameters. For example, the constrained optimization method of (Mozer et al., 2001) relies on approximations of sensitivity (which they call *CA*) and specificity[2] (their *CR*); related techniques (Gao et al., 2006; Jansche, 2005) rely on approximations of true positives, false positives, and false negatives, and, indirectly, recall and precision. Unlike these methods we compute the expected F-score exactly, without relying on *ad hoc* approximations of the true positives, etc.

Being able to efficiently compute the expected F-score is a prerequisite for maximizing it during decoding. More precisely, we compute the expectation of the function

$$y \mapsto F(z,y), \tag{2}$$

which is a unary function obtained by holding the first argument of the binary function F fixed. It will henceforth be abbreviated as $F(z,\cdot)$, and we will denote its expected value by

$$\mathrm{E}[F(z,\cdot)] = \sum_{y \in \{0,1\}^{|z|}} F(z,y)\ \Pr(y). \tag{3}$$

[2] Defined as $[(\vec{1}-z)\cdot(\vec{1}-y)]/[(\vec{1}-y)\cdot(\vec{1}-y)]$.

This expectation is taken with respect to a probability model over binary label sequences, written as $\Pr(y)$ for simplicity. This probability model may be conditional, that is, in general it will depend on covariates x and parameters θ. We have suppressed both in our notation, since x is fixed during training and decoding, and we assume that the model is fully identified during decoding. This is for clarity only and does not limit the class of models, though we will introduce additional, limiting assumptions shortly. We are now ready to tackle the inference task formally.

2 Maximum Expected *F*-Score Inference

2.1 Problem Statement

Optimal predictive inference under F-score utility requires us to find an hypothesis $\hat{z}$ of length n which maximizes the expected F-score relative to a given probabilistic sequence labeling model:

$$\hat{z} = \operatorname*{argmax}_{z \in \{0,1\}^n} \mathrm{E}[F(z,\cdot)] = \operatorname*{argmax}_{z \in \{0,1\}^n} \sum_y F(z,y)\ \Pr(y). \tag{4}$$

We require the probability model to factor into independent Bernoulli components (Markov order zero):

$$\Pr(y = (y_1,\ldots,y_n)) = \prod_{i=1}^{n} p_i^{y_i}\,(1-p_i)^{1-y_i}. \tag{5}$$

In practical applications we might choose the overall probability distribution to be the product of independent logistic regression models, for example. Ordinary classification arises as a special case when the y_i are i.i.d., that is, a single probabilistic classifier is used to find $\Pr(y_i = 1 \mid x_i)$. For our present purposes it is sufficient to assume that the inference algorithm takes as its input the vector $(p_1,\ldots,p_n)$, where p_i is the probability that $y_i = 1$.

The discrete maximization problem (4) cannot be solved naively, since the number of hypotheses that would need to be evaluated in a brute-force search for an optimal hypothesis $\hat{z}$ is exponential in the sequence length n. We show below that in fact only a few hypotheses ($n+1$ instead of 2^n) need to be examined in order to find an optimal one.

The inference algorithm is the intuitive one, analogous to the following simple observation: Start with the hypothesis $z = 00\ldots0$ and evaluate its raw F-score $F(z,y)$ relative to a fixed but unknown binary

string y. Then z will have perfect precision (no positive labels means no chance to make mistakes), and zero recall (unless $y = z$). Switch on any bit of z that is currently off. Then precision will decrease or remain equal, while recall will increase or remain equal. Repeat until $z = 11\ldots1$ is reached, in which case recall will be perfect and precision at its minimum. The inference algorithm for expected F-score follows the same strategy, and in particular it switches on the bits of z in order of non-increasing probability: start with $00\ldots0$, then switch on the bit $i_1 = \operatorname{argmax}_i p_i$, etc. until $11\ldots1$ is reached. We now show that this intuitive strategy is indeed admissible.

2.2 Outer and Inner Maximization

In general, maximization can be carried out piecewise, since

$$\operatorname*{argmax}_{x\in X} f(x) = \operatorname*{argmax}_{x\in\{\operatorname{argmax}_{y\in Y} f(y)\mid Y\in\pi(X)\}} f(x),$$

where $\pi(X)$ is any family $(Y_1, Y_2, \ldots)$ of nonempty subsets of X whose union $\bigcup_i Y_i$ is equal to X. (Recursive application would lead to a divide-and-conquer algorithm.) Duplication of effort is avoided if $\pi(X)$ is a partition of X.

Here we partition the set $\{0,1\}^n$ into equivalence classes based on the number of ones in a string (viewed as a real vector). Define S_m to be the set

$$S_m = \{s \in \{0,1\}^n \mid s \cdot s = m\}$$

consisting of all binary strings of fixed length n that contain exactly m ones. Then the maximization problem (4) can be transformed into an inner maximization

$$\hat{s}^{(m)} = \operatorname*{argmax}_{s\in S_m} \mathrm{E}\left[F(s,\cdot)\right], \tag{6}$$

followed by an outer maximization

$$\hat{z} = \operatorname*{argmax}_{z\in\{\hat{s}^{(0)},\ldots,\hat{s}^{(n)}\}} \mathrm{E}\left[F(z,\cdot)\right]. \tag{7}$$

2.3 Closed-Form Inner Maximization

The key insight is that the inner maximization problem (6) can be solved analytically. Given a vector $p = (p_1, \ldots, p_n)$ of probabilities, define $z^{(m)}$ to be the binary label sequence with exactly m ones and $n-m$ zeroes where for all indices i, k we have

$$\left[z_i^{(m)} = 1 \wedge z_k^{(m)} = 0\right] \rightarrow p_i \geq p_k.$$

Algorithm 1 Maximizing the Expected F-Score.

```
1: Input: probabilities p = (p_1, ..., p_n)
2: I ← indices of p sorted by non-increasing probability
3: z ← 0...0
4: a ← 0
5: v ← expectF(z, p)
6: for j ← 1 to n do
7:     i ← I[j]
8:     z[i] ← 1        // switch on the ith bit
9:     u ← expectF(z, p)
10:    if u > v then
11:        a ← j
12:        v ← u
13: for j ← a + 1 to n do
14:    z[I[j]] ← 0
15: return (z, v)
```

In other words, the most probable m bits (according to p) in $z^{(m)}$ are set and the least probable $n-m$ bits are off. We rely on the following result, whose proof is deferred to Appendix A:

Theorem 1. $(\forall s \in S_m)\ \mathrm{E}\left[F(z^{(m)},\cdot)\right] \geq \mathrm{E}\left[F(s,\cdot)\right]$.

Because $z^{(m)}$ is maximal in S_m, we may equate $z^{(m)} = \operatorname{argmax}_{s\in S_m} \mathrm{E}\left[F(s,\cdot)\right] = \hat{s}^{(m)}$ (modulo ties, which can always arise with argmax).

2.4 Pedestrian Outer Maximization

With the inner maximization (6) thus solved, the outer maximization (7) can be carried out naively, since only $n+1$ hypotheses need to be evaluated. This is precisely what Algorithm 1 does, which keeps track of the maximum value in v. On termination $z = \operatorname{argmax}_s \mathrm{E}\left[F(s,\cdot)\right]$. Correctness follows directly from our results in this section.

Algorithm 1 runs in time $\mathrm{O}(n \log n + n\, f(n))$. A total of $\mathrm{O}(n \log n)$ time is required for accessing the vector p in sorted order (line 2). This dominates the $\mathrm{O}(n)$ time required to explicitly generate the optimal hypothesis (lines 13–14). The algorithm invokes a subroutine *expectF*(z, p) a total of $n+1$ times. This subroutine, which is the topic of the next section, evaluates, in time $f(n)$, the expected F-score (with respect to p) of a given hypothesis z of length n.

3 Computing the Expected F-Score

3.1 Problem Statement

We now turn to the problem of computing the expected value (3) of the F-score for a given hypothesis z relative to a fully identified probability model. The method presented here does not strictly require the

zeroth-order Markov assumption (5) instated earlier (a higher-order Markov assumption will suffice), but it shall remain in effect for simplicity.

As with the maximization problem (4), the sum in (3) is over exponentially many terms and cannot be computed naively. But observe that the F-score (1) is a (rational) function of integer counts which are bounded, so it can take on only a finite, and indeed small, number of distinct values. We shall see shortly that the function (2) whose expectation we wish to compute has a domain whose cardinality is exponential in n, but the cardinality of its range is polynomial in n. The latter is sufficient to ensure that its expectation can be computed in polynomial time. The method we are about to develop is in fact very general and applies to many other loss and utility functions besides the F-score.

3.2 Expected F-Score as an Integral

A few notions from real analysis are helpful because they highlight the importance of thinking about functions in terms of their range, level sets, and the equivalence classes they induce on their domain (the kernel of the function). A function $g : \Omega \to \mathbb{R}$ is said to be *simple* if it can be expressed as a linear combination of indicator functions (characteristic functions):

$$g(x) = \sum_{k \in K} a_k \, \chi_{B_k}(x),$$

where K is a finite index set, $a_k \in \mathbb{R}$, and $B_k \subseteq \Omega$. ($\chi_S : S \to \{0,1\}$ is the characteristic function of set S.)

Let Ω be a countable set and $\mathscr{P}$ be a probability measure on Ω. Then the expectation of g is given by the Lebesgue integral of g. In the case of a simple function g as defined above, the integral, and hence the expectation, is defined as

$$\mathrm{E}[g] = \int_\Omega g \, \mathrm{d}\mathscr{P} = \sum_{k \in K} a_k \, \mathscr{P}(B_k). \qquad (8)$$

This gives us a general recipe for evaluating $\mathrm{E}[g]$ when Ω is much larger than the range of g. Instead of computing the sum $\sum_{y \in \Omega} g(y) \, \mathscr{P}(\{y\})$ we can compute the sum in (8) above. This directly yields an efficient algorithm whenever K is sufficiently small and $\mathscr{P}(B_k)$ can be evaluated efficiently.

The expected F-score is thus the Lebesgue integral of the function (2). Looking at the definition of the F-score in (1) we see that the only expressions which depend on y are $A = z \cdot y$ and $T = y \cdot y$ ($P = z \cdot z$ is fixed because z is). But $0 \le z \cdot y \le y \cdot y \le n = |z|$. Therefore $F(z, \cdot)$ takes on at most $(n+1)(n+2)/2$, i.e. quadratically many, distinct values. It is a simple function with

$$K = \{(A,T) \in \mathbb{N}_0 \times \mathbb{N}_0 \mid A \le T \le |z|,\ A \le z \cdot z\}$$

$$a_{(A,T)} = \frac{(\beta+1)\,A}{z \cdot z + \beta\, T} \qquad \text{where } 0/0 \stackrel{\text{def}}{=} 1$$

$$B_{(A,T)} = \{y \mid z \cdot y = A,\ y \cdot y = T\}.$$

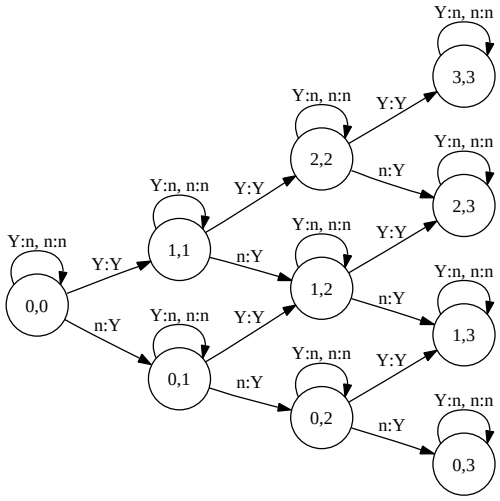


Figure 1: Finite State Classifier h'.

3.3 Computing Membership in B_k

Observe that the family of sets $\left(B_{(A,T)}\right)_{(A,T) \in K}$ is a partition (namely the kernel of $F(z,\cdot)$) of the set $\Omega = \{0,1\}^n$ of all label sequences of length n. In turn it gives rise to a function $h : \Omega \to K$ where $h(y) = k$ iff $y \in B_k$. The function h can be computed by a deterministic finite automaton, viewed as a sequence classifier: rather than assigning binary accept/reject labels, it assigns arbitrary labels from a finite set, in this case the index set K. For simplicity we show the initial portion of a slightly more general two-tape automaton h' in Figure 1. It reads the two sequences z and y on its two input tapes and counts the number of matching positive labels (represented as Y) as well as the number of positive labels on the second tape. Its behavior is therefore $h'(z,y) = (z \cdot y,\ y \cdot y)$. The function h is obtained as a special case when z (the first tape) is fixed.

Note that this only applies to the special case when

Algorithm 2 Simple Function Instance for F-Score.

```
def start():
  return (0,0)
def transition(k,z,i,y_i):
  (A,T) ← k
  if y_i = 1 then
    T ← T+1
    if z[i] = 1 then
      A ← A+1
  return (A,T)
def a(k,z):
  (A,T) ← k
  F ← (β+1)A / (z·z+βT)        // where 0/0 ≝ 1
  return F
```

Algorithm 3 Value of a Simple Function.

```
1: Input: instance g of the simple function interface, strings z
   and y of length n
2: k ← g.start()
3: for i ← 1 to n do
4:     k ← g.transition(k,z,i,y[i])
5: return g.a(k,z)
```

the family $B = (B_k)_{k \in K}$ is a partition of Ω. It is always possible to express any simple function in this way, but in general there may be an exponential increase in the size of K when the family B is required to be a partition. However for the special cases we consider here this problem does not arise.

3.4 The Simple Function Trick

In general, what we will call the *simple function trick* amounts to representing the simple function g whose expectation we want to compute by:

1. a finite index set K (perhaps implicit),
2. a deterministic finite state classifier $h : \Omega \to K$,
3. and a vector of coefficients $(a_k)_{k \in K}$.

In practice, this means instantiating an interface with three methods: the start and transition function of the transducer which computes h' (and from which h can be derived), and an accessor method for the coefficients a. Algorithm 2 shows the F-score instance.

Any simple function g expressed as an instance of this interface can then be evaluated very simply as $g(x) = a_{h(x)}$. This is shown in Algorithm 3.

Evaluating $\mathrm{E}[g]$ is also straightforward: Compose the DFA h with the probability model p and use an algebraic path algorithm to compute the total probability mass $\mathscr{P}(B_k)$ for each final state k of the resulting automaton. If p factors into independent components as required by (5), the composition is greatly simplified. If p incorporates label history (higher-order Markov assumption), nothing changes in principle, though the following algorithm assumes for simplicity that the stronger assumption is in effect.

Algorithm 4 Expectation of a Simple Function.

```
 1: Input: instance g of the simple function interface, string z
    and probability vector p of length n
 2: M ← Map()
 3: M[g.start()] ← 1
 4: for i ← 1 to n do
 5:     N ← Map()
 6:     for (k,P) ∈ M do
 7:         // transition on y_i = 0
 8:         k_0 ← g.transition(k,z,i,0)
 9:         if k_0 ∉ N then
10:             N[k_0] ← 0
11:         N[k_0] ← N[k_0] + P × (1 − p[i])
12:         // transition on y_i = 1
13:         k_1 ← g.transition(k,z,i,1)
14:         if k_1 ∉ N then
15:             N[k_1] ← 0
16:         N[k_1] ← N[k_1] + P × p[i]
17:     M ← N
18: E ← 0
19: for (k,P) ∈ M do
20:     E ← E + g.a(k,z) × P
21: return E
```

Algorithm 4 expands the following composed automaton, represented implicitly: the finite-state transducer h' specified as part of the simple function object g is composed on the left with the string z (yielding h) and on the right with the probability model p. The outer loop variable i is an index into z and hence a state in the automaton that accepts z; the variable k keeps track of the states of the automaton implemented by g; and the probability model has a single state by assumption, which does not need to be represented explicitly. Exploring the states in order of increasing i puts them in topological order, which means that the algebraic path problem can be solved in time linear in the size of the composed automaton. The maps M and N keep track of the algebraic distance from the start state to each intermediate state. On termination of the first outer loop (lines 4–17), the map M contains the final states together with their distances. The algebraic distance of a final state k is now equal to $\mathscr{P}(B_k)$, so the expected value E can be computed in the second loop (lines 18–20) as suggested by (8).

When the utility function interface g is instantiated as in Algorithm 2 to represent the F-score, the runtime of Algorithm 4 is cubic in n, with very small

constants.[3] The first main loop iterates over n. The inner loop iterates over the states expanded at iteration i, of which there are $O(i^2)$ many when dealing with the F-score. The second main loop iterates over the final states, whose number is quadratic in n in this case. The overall cubic runtime of the first loop dominates the computation.

3.5 Other Utility Functions

With other functions g the runtime of Algorithm 4 will depend on the asymptotic size of the index set K. If there are asymptotically as many intermediate states at any point as there are final states, then the general asymptotic runtime is $O(n|K|)$.

Many loss/utility functions are subsumed by the present framework. Zero–one loss is trivial: the automaton has two states (success, failure); it starts and remains in the success state as long as the symbols read on both tapes match; on the first mismatch it transitions to, and remains in, the failure state.

Hamming (1950) distance is similar to zero–one loss, but counts the number of mismatches (bounded by n), whereas zero–one loss only counts up to a threshold of one.

A more interesting case is given by the P_k-score (Beeferman et al., 1999) and its generalizations, which moves a sliding window of size k over a pair of label sequences (z, y) and counts the number of windows which contain a segment boundary on one of the sequences but not the other. To compute its expectation in our framework, all we have to do is express the sliding window mechanism as an automaton, which can be done very naturally (see the proof-of-concept implementation for further details).

4 Faster Inexact Computations

Because the exact computation of the expected F-score by Algorithm 4 requires cubic time, the overall runtime of Algorithm 1 (the decoder) is quartic.[4]

Faster decoding can be achieved by modifying Algorithm 4 to compute an approximation (in fact, a lower bound) of the expected F-score.[5] This is done by introducing an additional parameter L which limits the number of intermediate states that get expanded. Instead of iterating over all states and their associated probabilities (inner loop starting at line 6), one iterates over the top L states only. We require that $L \geq 1$ for this to be meaningful. Before entering the inner loop the entries of the map M are expanded and, using the linear time selection algorithm, the top L entries are selected. Because each state that gets expanded in the inner loop has out-degree 2, the new state map N will contain at most $2L$ states. This means that we have an additional loop invariant: the size of M is always less than or equal to $2L$. Therefore the selection algorithm runs in time $O(L)$, and so does the abridged inner loop, as well as the second outer loop. The overall runtime of this modified algorithm is therefore $O(nL)$.

If L is a constant function, the inexact computation of the expected F-score runs in linear time and the overall decoding algorithm in quadratic time. In particular if $L = 1$ the approximate expected F-score is equal to the F-score of the MAP hypothesis, and the modified inference algorithm reduces to a variant of Viterbi decoding. If L is a linear function of n, the overall decoding algorithm runs in cubic time.

We experimentally compared the exact quartic-time decoding algorithm with the approximate decoding algorithm for $L = 2n$ and for $L = 1$. We computed the absolute difference between the expected F-score of the optimal hypothesis (as found by the exact algorithm) and the expected F-score of the winning hypothesis found by the approximate decoding algorithm. For different sequence lengths $n \in \{1, \ldots, 50\}$ we performed 10 runs of the different decoding algorithms on randomly generated probability vectors p, where each p_i was randomly drawn from a continuous uniform distribution on $(0, 1)$, or, in a second experiment, from a Beta$(1/2, 1/2)$ distribution (to simulate an over-trained classifier).

For $L = 1$ there is a substantial difference of about

[3] A tight upper bound on the total number of states of the composed automaton in the worst case is $\left\lfloor \frac{1}{12}n^3 + \frac{5}{8}n^2 + \frac{17}{12}n + 1 \right\rfloor$.

[4] It is possible to speed up the decoding algorithm in absolute terms, though not asymptotically, by exploiting the fact that it explores very similar hypotheses in sequence. Algorithm 4 can be modified to store and return all of its intermediate map data-structures. This modified algorithm then requires cubic space instead of quadratic space. This additional storage cost pays off when the algorithm is called a second time, with its formal parameter z bound to a string that differs from the one of the preceding run in just one position. This means that the map data-structures only need to be recomputed from that position forward. However, this does not lead to an asymptotically faster algorithm in the worst case.

[5] For error bounds, see the proof-of-concept implementation.

0.6 between the expected F-scores of the winning hypothesis computed by the exact algorithm and by the approximate algorithm. Nevertheless the approximate decoding algorithm found the optimal hypothesis more than 99% of the time. This is presumably due to the additional regularization inherent in the discrete maximization of the decoder proper: even though the computed expected F-scores may be far from their exact values, this does not necessarily affect the behavior of the decoder very much, since it only needs to find the maximum among a small number of such scores. The error introduced by the approximation would have to be large enough to disturb the order of the hypotheses examined by the decoder in such a way that the true maximum is reordered. This generally does not seem to happen.

For $L = 2n$ the computed approximate expected F-scores were indistinguishable from their exact values. Consequently the approximate decoder found the true maximum every time.

5 Conclusion and Related Work

We have presented efficient algorithms for maximum expected F-score decoding. Our exact algorithm runs in quartic time, but an approximate cubic-time variant is indistinguishable in practice. A quadratic-time approximation makes very few mistakes and remains practically useful.

We have further described a general framework for computing the expectations of certain loss/utility functions. Our method relies on the fact that many functions are sparse, in the sense of having a finite range that is much smaller than their codomain. To evaluate their expectations, we can use the simple function trick and concentrate on their level sets: it suffices to evaluate the probability of those sets/ events. The fact that the commonly used utility functions like the F-score have only polynomially many level sets is sufficient (but not necessary) to ensure that our method is efficient. Because the coefficients a_k can be arbitrary (in fact, they can be generalized to be elements of a vector space over the reals), we can deal with functions that go beyond simple counts.

Like the methods developed by Allauzen et al. (2003) and Cortes et al. (2003) our technique incorporates finite automata, but uses a direct threshold-counting technique, rather than a nondeterministic counting technique which relies on path multiplicities. This makes it easy to formulate the simultaneous counting of two distinct quantities, such as our A and T, and to reason about the resulting automata.

The method described here is similar in spirit to those of Gao et al. (2006) and Jansche (2005), who discuss maximum expected F-score training of decision trees and logistic regression models. However, the present work is considerably more general in two ways: (1) the expected utility computations presented here are not tied in any way to particular classifiers, but can be used with large classes of probabilistic models; and (2) our framework extends beyond the computation of F-scores, which fall out as a special case, to other loss and utility functions, including the P_k score. More importantly, expected F-score computation as presented here can be exact, if desired, whereas the cited works always use an approximation to the quantities we have called A and T.

Acknowledgements

Most of this research was conducted while I was affilated with the Center for Computational Learning Systems, Columbia University. I would like to thank my colleagues at Google, in particular Ryan McDonald, as well as two anonymous reviewers for valuable feedback.

References

Cyril Allauzen, Mehryar Mohri, and Brian Roark. 2003. Generalized algorithms for constructing language models. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*.

Doug Beeferman, Adam Berger, and John Lafferty. 1999. Statistical models for text segmentation. *Machine Learning*, 34(1–3):177–210.

Francisco Casacuberta and Colin de la Higuera. 2000. Computational complexity of problems on probabilistic grammars and transducers. In *5th International Colloquium on Grammatical Inference*.

Corinna Cortes, Patrick Haffner, and Mehryar Mohri. 2003. Rational kernels. In *Advances in Neural Information Processing Systems*, volume 15.

Sheng Gao, Wen Wu, Chin-Hui Lee, and Tai-Seng Chua. 2006. A maximal figure-of-merit (MFoM)-learning approach to robust classifier design for text categorization. *ACM Transactions on Information Systems*, 24(2):190–218. Also in ICML 2004.

Samuel S. Gross, Olga Russakovsky, Chuong B. Do, and Serafim Batzoglou. 2007. Training conditional random fields for maximum labelwise accuracy. In *Advances in Neural Information Processing Systems*, volume 19.

R. W. Hamming. 1950. Error detecting and error correcting codes. *The Bell System Technical Journal*, 26(2):147–160.

Martin Jansche. 2005. Maximum expected F-measure training of logistic regression models. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615.

Michael C. Mozer, Robert Dodier, Michael D. Colagrosso, César Guerra-Salcedo, and Richard Wolniewicz. 2001. Prodding the ROC curve: Constrained optimization of classifier performance. In *Advances in Neural Information Processing Systems*, volume 14.

David R. Musicant, Vipin Kumar, and Aysel Ozgur. 2003. Optimizing F-measure with support vector machines. In *Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference*.

Jun Suzuki, Erik McDermott, and Hideki Isozaki. 2006. Training conditional random fields with multivariate evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*.

C. J. van Rijsbergen. 1974. Foundation of evaluation. *Journal of Documentation*, 30(4):365–373.

Appendix A Proof of Theorem 1

The proof of Theorem 1 employs the following lemma:

Theorem 2. *For fixed n and p, let $s,t \in S_m$ for some m with $1 \leq m < n$. Further assume that s and t differ only in two bits, i and k, in such a way that $s_i = 1$, $s_k = 0$; $t_i = 0$, $t_k = 1$; and $p_i \geq p_k$. Then $\mathrm{E}[F(s,\cdot)] \geq \mathrm{E}[F(t,\cdot)]$.*

Proof. Express the expected F-score $\mathrm{E}[F(s,\cdot)]$ as a sum and split the summation into two parts:

$$\sum_y F(s,y)\Pr(y) = \sum_{\substack{y \\ y_i = y_k}} F(s,y)\Pr(y) + \sum_{\substack{y \\ y_i \neq y_k}} F(s,y)\Pr(y).$$

If $y_i = y_k$ then $F(s,y) = F(t,y)$, for three reasons: the number of ones in s and t is the same (namely m) by assumption; y is constant; and the number of true positives is the same, that is $s \cdot y = t \cdot y$. The latter holds because s and y agree everywhere except on i and k; if $y_i = y_k = 0$, then there are no true positives at i and k; and if $y_i = y_k = 1$ then s_i is a true positive but s_k is not, and conversely t_k is but t_i is not. Therefore

$$\sum_{\substack{y \\ y_i = y_k}} F(s,y)\Pr(y) = \sum_{\substack{y \\ y_i = y_k}} F(t,y)\Pr(y). \tag{9}$$

Focus on those summands where $y_i \neq y_k$. Specifically group them into pairs (y,z) where y and z are identical except that $y_i = 1$ and $y_k = 0$, but $z_i = 0$ and $z_k = 1$. In other words, the two summations on the right-hand side of the following equality are carried out in parallel:

$$\sum_{\substack{y \\ y_i \neq y_k}} F(s,y)\Pr(y) = \sum_{\substack{y \\ y_i = 1 \\ y_k = 0}} F(s,y)\Pr(y) + \sum_{\substack{z \\ z_i = 0 \\ z_k = 1}} F(s,z)\Pr(z).$$

Then, focusing on s first:

$$\begin{aligned}
&F(s,y)\Pr(y) + F(s,z)\Pr(z) \\
&= \frac{(\beta+1)(A+1)}{m+\beta T}\Pr(y) + \frac{(\beta+1)A}{m+\beta T}\Pr(z) \\
&= \left[(A+1)p_i(1-p_k) + A(1-p_i)p_k\right]\frac{(\beta+1)}{m+\beta T}\,C \\
&= \left[p_i + (p_i + p_k - 2p_ip_k)A - p_ip_k\right]\frac{(\beta+1)}{m+\beta T}\,C \\
&= [p_i + C_0]\,C_1,
\end{aligned}$$

where $A = s \cdot z$ is the number of true positives between s and z (s and y have an additional true positive at i by construction); $T = y \cdot y = z \cdot z$ is the number of positive labels in y and z (identical by assumption); and

$$C = \frac{\Pr(y)}{p_i(1-p_k)} = \frac{\Pr(z)}{(1-p_i)p_k}$$

is the probability of y and z evaluated on all positions except for i and k. This equality holds because of the zeroth-order Markov assumption (5) imposed on $\Pr(y)$. C_0 and C_1 are constants that allow us to focus on the essential aspects.

The situation for t is similar, except for the true positives:

$$\begin{aligned}
&F(t,y)\Pr(y) + F(t,z)\Pr(z) \\
&= \frac{(\beta+1)A}{m+\beta T}\Pr(y) + \frac{(\beta+1)(A+1)}{m+\beta T}\Pr(z) \\
&= \left[A\,p_i(1-p_k) + (A+1)(1-p_i)p_k\right]\frac{(\beta+1)}{m+\beta T}\,C \\
&= \left[p_k + (p_i + p_k - 2p_ip_k)A - p_ip_k\right]\frac{(\beta+1)}{m+\beta T}\,C \\
&= [p_k + C_0]\,C_1
\end{aligned}$$

where all constants have the same values as above. But $p_i \geq p_k$ by assumption, $p_k + C_0 \geq 0$, and $C_1 \geq 0$, so we have

$$\begin{aligned}
&F(s,y)\Pr(y) + F(s,z)\Pr(z) = [p_i + C_0]\,C_1 \\
&\geq F(t,y)\Pr(y) + F(t,z)\Pr(z) = [p_k + C_0]\,C_1,
\end{aligned}$$

and therefore

$$\sum_{\substack{y \\ y_i \neq y_k}} F(s,y)\Pr(y) \geq \sum_{\substack{y \\ y_i \neq y_k}} F(t,y)\Pr(y). \tag{10}$$

The theorem follows from equality (9) and inequality (10). □

Proof of Theorem 1: $(\forall s \in S_m)\ \mathrm{E}[F(z^{(m)},\cdot)] \geq \mathrm{E}[F(s,\cdot)]$. Observe that $z^{(m)} \in S_m$ by definition (see Section 2.3). For $m = 0$ and $m = n$ the theorem holds trivially because S_m is a singleton set. In the nontrivial cases, Theorem 2 is applied repeatedly. The string $z^{(m)}$ can be transformed into any other string $s \in S_m$ by repeatedly clearing a more likely set bit and setting a less likely unset bit.

In particular this can be done as follows: First, find the indices where $z^{(m)}$ and s disagree. By construction there must be an even number of such indices; indeed there are equinumerous sets

$$\left\{ i \mid z_i^{(m)} = 1 \wedge s_i = 0 \right\} \approx \left\{ j \mid z_j^{(m)} = 0 \wedge s_j = 1 \right\}.$$

This holds because the total number of ones is fixed and identical in $z^{(m)}$ and s, and so is the total number of zeroes. Next, sort those indices by non-increasing probability and represent them as $i_1, \ldots, i_k$ and $j_1, \ldots, j_k$. Let $s_0 = z^{(m)}$. Then let s_1 be identical to s_0 except that $s_{i_1} = 0$ and $s_{j_1} = 1$. Form $s_2, \ldots, s_k$ along the same lines and observe that $s_k = s$ by construction. By definition of $z^{(m)}$ it must be the case that $p_{i_r} \geq p_{j_r}$ for all $r \in \{1, \ldots, k\}$. Therefore Theorem 2 applies at every step along the way from $z^{(m)} = s_0$ to $s_k = s$, and so the expected utility is non-increasing along that path. □

A Fully Bayesian Approach to Unsupervised Part-of-Speech Tagging*

Sharon Goldwater
Department of Linguistics
Stanford University
sgwater@stanford.edu

Thomas L. Griffiths
Department of Psychology
UC Berkeley
tom_griffiths@berkeley.edu

Abstract

Unsupervised learning of linguistic structure is a difficult problem. A common approach is to define a generative model and maximize the probability of the hidden structure given the observed data. Typically, this is done using maximum-likelihood estimation (MLE) of the model parameters. We show using part-of-speech tagging that a fully Bayesian approach can greatly improve performance. Rather than estimating a single set of parameters, the Bayesian approach integrates over all possible parameter values. This difference ensures that the learned structure will have high probability over a range of possible parameters, and permits the use of priors favoring the sparse distributions that are typical of natural language. Our model has the structure of a standard trigram HMM, yet its accuracy is closer to that of a state-of-the-art discriminative model (Smith and Eisner, 2005), up to 14 percentage points better than MLE. We find improvements both when training from data alone, and using a tagging dictionary.

1 Introduction

Unsupervised learning of linguistic structure is a difficult problem. Recently, several new model-based approaches have improved performance on a variety of tasks (Klein and Manning, 2002; Smith and Eisner, 2005). Nearly all of these approaches have one aspect in common: the goal of learning is to identify the set of model parameters that maximizes some objective function. Values for the hidden variables in the model are then chosen based on the learned parameterization. Here, we propose a different approach based on Bayesian statistical principles: rather than searching for an optimal set of parameter values, we seek to directly maximize the probability of the hidden variables given the observed data, integrating over all possible parameter values. Using part-of-speech (POS) tagging as an example application, we show that the Bayesian approach provides large performance improvements over maximum-likelihood estimation (MLE) for the same model structure. Two factors can explain the improvement. First, integrating over parameter values leads to greater robustness in the choice of tag sequence, since it must have high probability over a range of parameters. Second, integration permits the use of priors favoring sparse distributions, which are typical of natural language. These kinds of priors can lead to degenerate solutions if the parameters are estimated directly.

Before describing our approach in more detail, we briefly review previous work on unsupervised POS tagging. Perhaps the most well-known is that of Merialdo (1994), who used MLE to train a trigram hidden Markov model (HMM). More recent work has shown that improvements can be made by modifying the basic HMM structure (Banko and Moore, 2004), using better smoothing techniques or added constraints (Wang and Schuurmans, 2005), or using a discriminative model rather than an HMM

*This work was supported by grants NSF 0631518 and ONR MURI N000140510388. We would also like to thank Noah Smith for providing us with his data sets.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 744–751,
Prague, Czech Republic, June 2007.

(Smith and Eisner, 2005). Non-model-based approaches have also been proposed (Brill (1995); see also discussion in Banko and Moore (2004)). All of this work is really POS *disambiguation*: learning is strongly constrained by a dictionary listing the allowable tags for each word in the text. Smith and Eisner (2005) also present results using a diluted dictionary, where infrequent words may have any tag. Haghighi and Klein (2006) use a small list of labeled prototypes and no dictionary.

A different tradition treats the identification of syntactic classes as a knowledge-free clustering problem. Distributional clustering and dimensionality reduction techniques are typically applied when linguistically meaningful classes are desired (Schütze, 1995; Clark, 2000; Finch et al., 1995); probabilistic models have been used to find classes that can improve smoothing and reduce perplexity (Brown et al., 1992; Saul and Pereira, 1997). Unfortunately, due to a lack of standard and informative evaluation techniques, it is difficult to compare the effectiveness of different clustering methods.

In this paper, we hope to unify the problems of POS disambiguation and syntactic clustering by presenting results for conditions ranging from a full tag dictionary to no dictionary at all. We introduce the use of a new information-theoretic criterion, *variation of information* (Meilă, 2002), which can be used to compare a gold standard clustering to the clustering induced from a tagger's output, regardless of the cluster labels. We also evaluate using tag accuracy when possible. Our system outperforms an HMM trained with MLE on both metrics in all circumstances tested, often by a wide margin. Its accuracy in some cases is close to that of Smith and Eisner's (2005) discriminative model. Our results show that the Bayesian approach is particularly useful when learning is less constrained, either because less evidence is available (corpus size is small) or because the dictionary contains less information.

In the following section, we discuss the motivation for a Bayesian approach and present our model and search procedure. Section 3 gives results illustrating how the parameters of the prior affect results, and Section 4 describes how to infer a good choice of parameters from unlabeled data. Section 5 presents results for a range of corpus sizes and dictionary information, and Section 6 concludes.

2 A Bayesian HMM

2.1 Motivation

In model-based approaches to unsupervised language learning, the problem is formulated in terms of identifying latent structure from data. We define a model with parameters θ, some observed variables $\mathbf{w}$ (the linguistic input), and some latent variables $\mathbf{t}$ (the hidden structure). The goal is to assign appropriate values to the latent variables. Standard approaches do so by selecting values for the model parameters, and then choosing the most probable variable assignment based on those parameters. For example, maximum-likelihood estimation (MLE) seeks parameters $\hat{\theta}$ such that

$$\hat{\theta} = \operatorname*{argmax}_{\theta} P(\mathbf{w}|\theta), \tag{1}$$

where $P(\mathbf{w}|\theta) = \sum_{\mathbf{t}} P(\mathbf{w}, \mathbf{t}|\theta)$. Sometimes, a non-uniform prior distribution over θ is introduced, in which case $\hat{\theta}$ is the *maximum a posteriori* (MAP) solution for θ:

$$\hat{\theta} = \operatorname*{argmax}_{\theta} P(\mathbf{w}|\theta)P(\theta). \tag{2}$$

The values of the latent variables are then taken to be those that maximize $P(\mathbf{t}|\mathbf{w}, \hat{\theta})$.

In contrast, the Bayesian approach we advocate in this paper seeks to identify a distribution over latent variables directly, without ever fixing particular values for the model parameters. The distribution over latent variables given the observed data is obtained by integrating over all possible values of θ:

$$P(\mathbf{t}|\mathbf{w}) = \int P(\mathbf{t}|\mathbf{w}, \theta)P(\theta|\mathbf{w})d\theta. \tag{3}$$

This distribution can be used in various ways, including choosing the MAP assignment to the latent variables, or estimating expected values for them.

To see why integrating over possible parameter values can be useful when inducing latent structure, consider the following example. We are given a coin, which may be biased ($t = 1$) or fair ($t = 0$), each with probability .5. Let θ be the probability of heads. If the coin is biased, we assume a uniform distribution over θ, otherwise $\theta = .5$. We observe $\mathbf{w}$, the outcomes of 10 coin flips, and we wish to determine whether the coin is biased (i.e. the value of

t). Assume that we have a uniform prior on θ, with $p(\theta) = 1$ for all $\theta \in [0, 1]$. First, we apply the standard methodology of finding the MAP estimate for θ and then selecting the value of t that maximizes $P(t|\mathbf{w}, \hat{\theta})$. In this case, an elementary calculation shows that the MAP estimate is $\hat{\theta} = n_H/10$, where n_H is the number of heads in $\mathbf{w}$ (likewise, n_T is the number of tails). Consequently, $P(t|\mathbf{w}, \hat{\theta})$ favors $t = 1$ for any sequence that does not contain exactly five heads, and assigns equal probability to $t = 1$ and $t = 0$ for any sequence that does contain exactly five heads — a counterintuitive result. In contrast, using some standard results in Bayesian analysis we can show that applying Equation 3 yields

$$P(t = 1|\mathbf{w}) = 1/\left(1 + \frac{11!}{n_H! n_T! 2^{10}}\right) \quad (4)$$

which is significantly less than .5 when $n_H = 5$, and only favors $t = 1$ for sequences where $n_H \geq 8$ or $n_H \leq 2$. This intuitively sensible prediction results from the fact that the Bayesian approach is sensitive to the *robustness* of a choice of t to the value of θ, as illustrated in Figure 1. Even though a sequence with $n_H = 6$ yields a MAP estimate of $\hat{\theta} = 0.6$ (Figure 1 (a)), $P(t = 1|\mathbf{w}, \theta)$ is only greater than 0.5 for a small range of θ around $\hat{\theta}$ (Figure 1 (b)), meaning that the choice of $t = 1$ is not very robust to variation in θ. In contrast, a sequence with $n_H = 8$ favors $t = 1$ for a wide range of θ around $\hat{\theta}$. By integrating over θ, Equation 3 takes into account the consequences of possible variation in θ.

Another advantage of integrating over θ is that it permits the use of linguistically appropriate priors. In many linguistic models, including HMMs, the distributions over variables are multinomial. For a multinomial with parameters $\theta = (\theta_1, \ldots, \theta_K)$, a natural choice of prior is the K-dimensional Dirichlet distribution, which is conjugate to the multinomial.[1] For simplicity, we initially assume that all K parameters (also known as *hyperparameters*) of the Dirichlet distribution are equal to β, i.e. the Dirichlet is *symmetric*. The value of β determines which parameters θ will have high probability: when $\beta = 1$, all parameter values are equally likely; when $\beta > 1$, multinomials that are closer to uniform are preferred; and when $\beta < 1$, high probability is assigned to sparse multinomials, where one or more parameters are at or near 0.

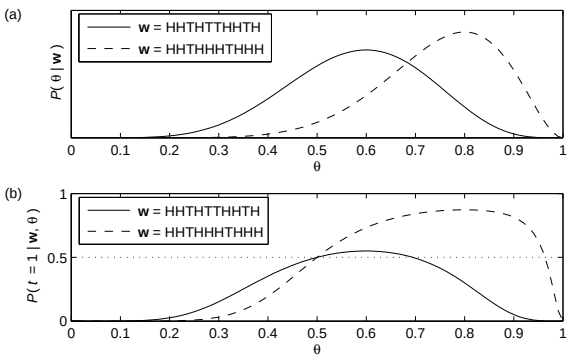


Figure 1: The Bayesian approach to estimating the value of a latent variable, t, from observed data, $\mathbf{w}$, chooses a value of t robust to uncertainty in θ. (a) Posterior distribution on θ given $\mathbf{w}$. (b) Probability that $t = 1$ given $\mathbf{w}$ and θ as a function of θ.

Typically, linguistic structures are characterized by sparse distributions (e.g., POS tags are followed with high probability by only a few other tags, and have highly skewed output distributions). Consequently, it makes sense to use a Dirichlet prior with $\beta < 1$. However, as noted by Johnson et al. (2007), this choice of β leads to difficulties with MAP estimation. For a sequence of draws $\mathbf{x} = (x_1, \ldots, x_n)$ from a multinomial distribution θ with observed counts $n_1, \ldots, n_K$, a symmetric Dirichlet(β) prior over θ yields the MAP estimate $\theta_k = \frac{n_k + \beta - 1}{n + K(\beta - 1)}$. When $\beta \geq 1$, standard MLE techniques such as EM can be used to find the MAP estimate simply by adding "pseudocounts" of size $\beta - 1$ to each of the expected counts n_k at each iteration. However, when $\beta < 1$, the values of θ that set one or more of the θ_k equal to 0 can have infinitely high posterior probability, meaning that MAP estimation can yield degenerate solutions. If, instead of estimating θ, we integrate over all possible values, we no longer encounter such difficulties. Instead, the probability that outcome x_i takes value k given previous outcomes $\mathbf{x}_{-i} = (x_1, \ldots, x_{i-1})$ is

$$\begin{aligned} P(k|\mathbf{x}_{-i}, \beta) &= \int P(k|\theta) P(\theta|\mathbf{x}_{-i}, \beta)\, d\theta \\ &= \frac{n_k + \beta}{i - 1 + K\beta} \end{aligned} \quad (5)$$

[1] A prior is *conjugate* to a distribution if the posterior has the same form as the prior.

where n_k is the number of times k occurred in $\mathbf{x}_{-i}$. See MacKay and Peto (1995) for a derivation.

2.2 Model Definition

Our model has the structure of a standard trigram HMM, with the addition of symmetric Dirichlet priors over the transition and output distributions:

$$\begin{aligned} t_i|t_{i-1}=t,t_{i-2}=t',\tau^{(t,t')} &\sim \text{Mult}(\tau^{(t,t')}) \\ w_i|t_i=t,\omega^{(t)} &\sim \text{Mult}(\omega^{(t)}) \\ \tau^{(t,t')}|\alpha &\sim \text{Dirichlet}(\alpha) \\ \omega^{(t)}|\beta &\sim \text{Dirichlet}(\beta) \end{aligned}$$

where t_i and w_i are the ith tag and word. We assume that sentence boundaries are marked with a distinguished tag. For a model with T possible tags, each of the transition distributions $\tau^{(t,t')}$ has T components, and each of the output distributions $\omega^{(t)}$ has W_t components, where W_t is the number of word types that are permissible outputs for tag t. We will use τ and ω to refer to the entire transition and output parameter sets. This model assumes that the prior over state transitions is the same for all histories, and the prior over output distributions is the same for all states. We relax the latter assumption in Section 4.

Under this model, Equation 5 gives us

$$P(t_i|\mathbf{t}_{-i},\alpha) = \frac{n_{(t_{i-2},t_{i-1},t_i)}+\alpha}{n_{(t_{i-2},t_{i-1})}+T\alpha} \quad (6)$$

$$P(w_i|t_i,\mathbf{t}_{-i},\mathbf{w}_{-i},\beta) = \frac{n_{(t_i,w_i)}+\beta}{n_{(t_i)}+W_{t_i}\beta} \quad (7)$$

where $n_{(t_{i-2},t_{i-1},t_i)}$ and $n_{(t_i,w_i)}$ are the number of occurrences of the trigram (t_{i-2},t_{i-1},t_i) and the tag-word pair (t_i,w_i) in the $i-1$ previously generated tags and words. Note that, by integrating out the parameters τ and ω, we induce dependencies between the variables in the model. The probability of generating a particular trigram tag sequence (likewise, output) depends on the number of times that sequence (output) has been generated previously. Importantly, trigrams (and outputs) remain *exchangeable*: the probability of a set of trigrams (outputs) is the same regardless of the order in which it was generated. The property of exchangeability is crucial to the inference algorithm we describe next.

2.3 Inference

To perform inference in our model, we use Gibbs sampling (Geman and Geman, 1984), a stochastic procedure that produces samples from the posterior distribution $P(\mathbf{t}|\mathbf{w},\alpha,\beta) \propto P(\mathbf{w}|\mathbf{t},\beta)P(\mathbf{t}|\alpha)$. We initialize the tags at random, then iteratively resample each tag according to its conditional distribution given the current values of all other tags. Exchangeability allows us to treat the current counts of the other tag trigrams and outputs as "previous" observations. The only complication is that resampling a tag changes the identity of three trigrams at once, and we must account for this in computing its conditional distribution. The sampling distribution for t_i is given in Figure 2.

In Bayesian statistical inference, multiple samples from the posterior are often used in order to obtain statistics such as the expected values of model variables. For POS tagging, estimates based on multiple samples might be useful if we were interested in, for example, the probability that two words have the same tag. However, computing such probabilities across all pairs of words does not necessarily lead to a consistent clustering, and the result would be difficult to evaluate. Using a single sample makes standard evaluation methods possible, but yields suboptimal results because the value for each tag is sampled from a distribution, and some tags will be assigned low-probability values. Our solution is to treat the Gibbs sampler as a stochastic search procedure with the goal of identifying the MAP tag sequence. This can be done using tempering (annealing), where a temperature of ϕ is equivalent to raising the probabilities in the sampling distribution to the power of $\frac{1}{\phi}$. As ϕ approaches 0, even a single sample will provide a good MAP estimate.

3 Fixed Hyperparameter Experiments

3.1 Method

Our initial experiments follow in the tradition begun by Merialdo (1994), using a tag dictionary to constrain the possible parts of speech allowed for each word. (This also fixes W_t, the number of possible words for tag t.) The dictionary was constructed by listing, for each word, all tags found for that word in the entire WSJ treebank. For the experiments in this section, we used a 24,000-word subset of the tree-

$$P(t_i|\mathbf{t}_{-i},\mathbf{w},\alpha,\beta) \propto \frac{n_{(t_i,w_i)}+\beta}{n_{t_i}+W_{t_i}\beta}\cdot\frac{n_{(t_{i-2},t_{i-1},t_i)}+\alpha}{n_{(t_{i-2},t_{i-1})}+T\alpha}\cdot\frac{n_{(t_{i-1},t_i,t_{i+1})}+I(t_{i-2}=t_{i-1}=t_i=t_{i+1})+\alpha}{n_{(t_{i-1},t_i)}+I(t_{i-2}=t_{i-1}=t_i)+T\alpha}$$
$$\cdot\frac{n_{(t_i,t_{i+1},t_{i+2})}+I(t_{i-2}=t_i=t_{i+2},t_{i-1}=t_{i+1})+I(t_{i-1}=t_i=t_{i+1}=t_{i+2})+\alpha}{n_{(t_i,t_{i+1})}+I(t_{i-2}=t_i,t_{i-1}=t_{i+1})+I(t_{i-1}=t_i=t_{i+1})+T\alpha}$$

Figure 2: Conditional distribution for t_i. Here, $\mathbf{t}_{-i}$ refers to the current values of all tags except for t_i, $I(.)$ is a function that takes on the value 1 when its argument is true and 0 otherwise, and all counts n_x are with respect to the tag trigrams and tag-word pairs in $(\mathbf{t}_{-i},\mathbf{w}_{-i})$.

bank as our unlabeled training corpus. 54.5% of the tokens in this corpus have at least two possible tags, with the average number of tags per token being 2.3. We varied the values of the hyperparameters α and β and evaluated overall tagging accuracy. For comparison with our Bayesian HMM (BHMM) in this and following sections, we also present results from the Viterbi decoding of an HMM trained using MLE by running EM to convergence (MLHMM). Where direct comparison is possible, we list the scores reported by Smith and Eisner (2005) for their conditional random field model trained using contrastive estimation (CRF/CE).[2]

For all experiments, we ran our Gibbs sampling algorithm for 20,000 iterations over the entire data set. The algorithm was initialized with a random tag assignment and a temperature of 2, and the temperature was gradually decreased to .08. Since our inference procedure is stochastic, our reported results are an average over 5 independent runs.

Results from our model for a range of hyperparameters are presented in Table 1. With the best choice of hyperparameters ($\alpha = .003, \beta = 1$), we achieve average tagging accuracy of 86.8%. This far surpasses the MLHMM performance of 74.5%, and is closer to the 90.1% accuracy of CRF/CE on the same data set using oracle parameter selection. The effects of α, which determines the probability of the transition distributions, are stronger than the effects of β, which determines the probability of the output distributions. The optimal value of .003 for α reflects the fact that the true transition probability matrix for this corpus is indeed sparse. As α grows larger, the model prefers more uniform transition probabilities, which causes it to perform worse. Although the true output distributions tend to be sparse as well, the level of sparseness depends on the tag (consider function words vs. content words in particular). Therefore, a value of β that accurately reflects the most probable output distributions for some tags may be a poor choice for other tags. This leads to the smaller effect of β, and suggests that performance might be improved by selecting a different β for each tag, as we do in the next section.

Value of α	Value of β .001	.003	.01	.03	.1	.3	1.0
.001	85.0	85.7	86.1	86.0	86.2	86.5	86.6
.003	85.5	85.5	85.8	86.6	86.7	86.7	**86.8**
.01	85.3	85.5	85.6	85.9	86.4	86.4	86.2
.03	85.9	85.8	86.1	86.2	86.6	86.8	86.4
.1	85.2	85.0	85.2	85.1	84.9	85.5	84.9
.3	84.4	84.4	84.6	84.4	84.5	85.7	85.3
1.0	83.1	83.0	83.2	83.3	83.5	83.7	83.9

Table 1: Percentage of words tagged correctly by BHMM as a function of the hyperparameters α and β. Results are averaged over 5 runs on the 24k corpus with full tag dictionary. Standard deviations in most cases are less than .5.

A final point worth noting is that even when $\alpha = \beta = 1$ (i.e., the Dirichlet priors exert no influence) the BHMM still performs much better than the MLHMM. This result underscores the importance of integrating over model parameters: the BHMM identifies a sequence of tags that have high proba-

[2]Results of CRF/CE depend on the set of features used and the contrast neighborhood. In all cases, we list the best score reported for any contrast neighborhood using trigram (but no spelling) features. To ensure proper comparison, all corpora used in our experiments consist of the same randomized sets of sentences used by Smith and Eisner. Note that training on sets of contiguous sentences from the beginning of the treebank consistently improves our results, often by 1-2 percentage points or more. MLHMM scores show less difference between randomized and contiguous corpora.

bility over a range of parameter values, rather than choosing tags based on the single best set of parameters. The improved results of the BHMM demonstrate that selecting a sequence that is robust to variations in the parameters leads to better performance.

4 Hyperparameter Inference

In our initial experiments, we experimented with different fixed values of the hyperparameters and reported results based on their optimal values. However, choosing hyperparameters in this way is time-consuming at best and impossible at worst, if there is no gold standard available. Luckily, the Bayesian approach allows us to automatically select values for the hyperparameters by treating them as additional variables in the model. We augment the model with priors over the hyperparameters (here, we assume an improper uniform prior), and use a single Metropolis-Hastings update (Gilks et al., 1996) to resample the value of each hyperparameter after each iteration of the Gibbs sampler. Informally, to update the value of hyperparameter α, we sample a proposed new value α' from a normal distribution with $\mu = \alpha$ and $\sigma = .1\alpha$. The probability of accepting the new value depends on the ratio between $P(\mathbf{t}|\mathbf{w}, \alpha)$ and $P(\mathbf{t}|\mathbf{w}, \alpha')$ and a term correcting for the asymmetric proposal distribution.

Performing inference on the hyperparameters allows us to relax the assumption that every tag has the same prior on its output distribution. In the experiments reported in the following section, we used two different versions of our model. The first version (BHMM1) uses a single value of β for all word classes (as above); the second version (BHMM2) uses a separate β_j for each tag class j.

5 Inferred Hyperparameter Experiments

5.1 Varying corpus size

In this set of experiments, we used the full tag dictionary (as above), but performed inference on the hyperparameters. Following Smith and Eisner (2005), we trained on four different corpora, consisting of the first 12k, 24k, 48k, and 96k words of the WSJ corpus. For all corpora, the percentage of ambiguous tokens is 54%-55% and the average number of tags per token is 2.3. Table 2 shows results for the various models and a random baseline (averaged over 5 random tag assignments). Hyperparameter inference leads to slightly lower scores than are obtained by oracle hyperparameter selection, but both versions of BHMM are still far superior to MLHMM for all corpus sizes. Not surprisingly, the advantages of BHMM are most pronounced on the smallest corpus: the effects of parameter integration and sensible priors are stronger when less evidence is available from the input. In the limit as corpus size goes to infinity, the BHMM and MLHMM will make identical predictions.

	Corpus size			
Accuracy	12k	24k	48k	96k
random	64.8	64.6	64.6	64.6
MLHMM	71.3	74.5	76.7	78.3
CRF/CE	86.2	88.6	88.4	89.4
BHMM1	85.8	85.2	83.6	85.0
BHMM2	85.8	84.4	85.7	85.8
$\sigma <$	.7	.2	.6	.2

Table 2: Percentage of words tagged correctly by the various models on different sized corpora. BHMM1 and BHMM2 use hyperparameter inference; CRF/CE uses parameter selection based on an unlabeled development set. Standard deviations (σ) for the BHMM results fell below those shown for each corpus size.

5.2 Varying dictionary knowledge

In unsupervised learning, it is not always reasonable to assume that a large tag dictionary is available. To determine the effects of reduced or absent dictionary information, we ran a set of experiments inspired by those of Smith and Eisner (2005). First, we collapsed the set of 45 treebank tags onto a smaller set of 17 (the same set used by Smith and Eisner). We created a full tag dictionary for this set of tags from the entire treebank, and also created several reduced dictionaries. Each reduced dictionary contains the tag information only for words that appear at least d times in the training corpus (the 24k corpus, for these experiments). All other words are fully ambiguous between all 17 classes. We ran tests with $d = 1, 2, 3, 5, 10$, and ∞ (i.e., knowledge-free syntactic clustering).

With standard accuracy measures, it is difficult to

	Value of d					
Accuracy	1	2	3	5	10	∞
random	69.6	56.7	51.0	45.2	38.6	
MLHMM	83.2	70.6	65.5	59.0	50.9	
CRF/CE	90.4	77.0	71.7			
BHMM1	86.0	76.4	71.0	64.3	58.0	
BHMM2	87.3	79.6	65.0	59.2	49.7	
$\sigma <$	.2	.8	.6	.3	1.4	
VI						
random	2.65	3.96	4.38	4.75	5.13	7.29
MLHMM	1.13	2.51	3.00	3.41	3.89	6.50
BHMM1	1.09	2.44	2.82	3.19	3.47	4.30
BHMM2	**1.04**	**1.78**	**2.31**	**2.49**	**2.97**	**4.04**
$\sigma <$	.02	.03	.04	.03	.07	.17
Corpus stats						
% ambig.	49.0	61.3	66.3	70.9	75.8	100
tags/token	1.9	4.4	5.5	6.8	8.3	17

Table 3: Percentage of words tagged correctly and variation of information between clusterings induced by the assigned and gold standard tags as the amount of information in the dictionary is varied. Standard deviations (σ) for the BHMM results fell below those shown in each column. The percentage of ambiguous tokens and average number of tags per token for each value of d is also shown.

evaluate the quality of a syntactic clustering when no dictionary is used, since cluster names are interchangeable. We therefore introduce another evaluation measure for these experiments, a distance metric on clusterings known as *variation of information* (Meilă, 2002). The variation of information (VI) between two clusterings C (the gold standard) and C' (the found clustering) of a set of data points is a sum of the amount of information lost in moving from C to C', and the amount that must be gained. It is defined in terms of entropy H and mutual information I: $VI(C, C') = H(C) + H(C') - 2I(C, C')$. Even when accuracy can be measured, VI may be more informative: two different tag assignments may have the same accuracy but different VI with respect to the gold standard if the *errors* in one assignment are less consistent than those in the other.

Table 3 gives the results for this set of experiments. One or both versions of BHMM outperform MLHMM in terms of tag accuracy for all values of d, although the differences are not as great as in earlier experiments. The differences in VI are more striking, particularly as the amount of dictionary information is reduced. When ambiguity is greater, both versions of BHMM show less confusion with respect to the true tags than does MLHMM, and BHMM2 performs the best in all circumstances. The confusion matrices in Figure 3 provide a more intuitive picture of the very different sorts of clusterings produced by MLHMM and BHMM2 when no tag dictionary is available. Similar differences hold to a lesser degree when a partial dictionary is provided. With MLHMM, different tokens of the same word type are usually assigned to the same cluster, but types are assigned to clusters more or less at random, and all clusters have approximately the same number of types (542 on average, with a standard deviation of 174). The clusters found by BHMM2 tend to be more coherent and more variable in size: in the 5 runs of BHMM2, the average number of types per cluster ranged from 436 to 465 (i.e., tokens of the same word are spread over fewer clusters than in MLHMM), with a standard deviation between 460 and 674. Determiners, prepositions, the possessive marker, and various kinds of punctuation are mostly clustered coherently. Nouns are spread over a few clusters, partly due to a distinction found between common and proper nouns. Likewise, modal verbs and the copula are mostly separated from other verbs. Errors are often sensible: adjectives and nouns are frequently confused, as are verbs and adverbs.

The kinds of results produced by BHMM1 and BHMM2 are more similar to each other than to the results of MLHMM, but the differences are still informative. Recall that BHMM1 learns a single value for β that is used for all output distributions, while BHMM2 learns separate hyperparameters for each cluster. This leads to different treatments of difficult-to-classify low-frequency items. In BHMM1, these items tend to be spread evenly among all clusters, so that all clusters have similarly sparse output distributions. In BHMM2, the system creates one or two clusters consisting entirely of very infrequent items, where the priors on these clusters strongly prefer uniform outputs, and all other clusters prefer extremely sparse outputs (and are more coherent than in BHMM1). This explains the difference in VI between the two systems, as well as the higher accuracy of BHMM1 for $d \geq 3$: the single β discourages placing low-frequency items in their own cluster, so they are more likely to be clustered with items that have sim-

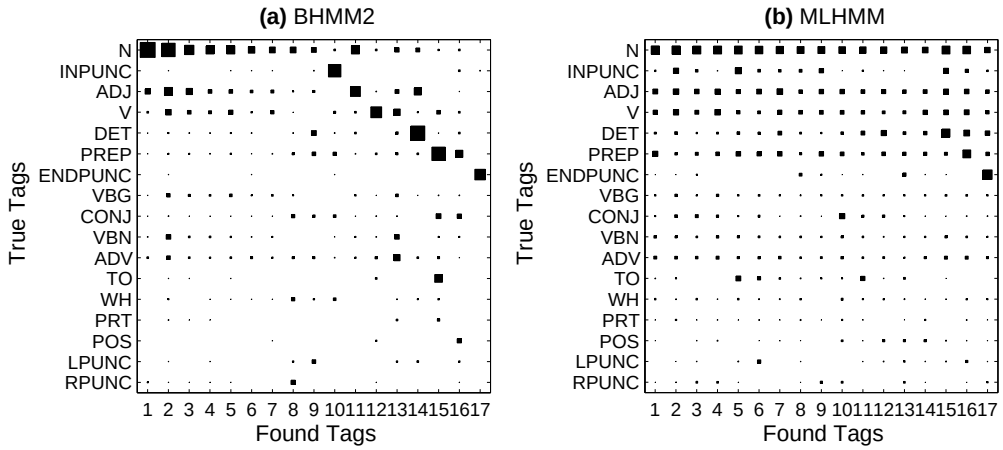


Figure 3: Confusion matrices for the dictionary-free clusterings found by (a) BHMM2 and (b) MLHMM.

ilar transition probabilities. The problem of junk clusters in BHMM2 might be alleviated by using a non-uniform prior over the hyperparameters to encourage some degree of sparsity in all clusters.

6 Conclusion

In this paper, we have demonstrated that, for a standard trigram HMM, taking a Bayesian approach to POS tagging dramatically improves performance over maximum-likelihood estimation. Integrating over possible parameter values leads to more robust solutions and allows the use of priors favoring sparse distributions. The Bayesian approach is particularly helpful when learning is less constrained, either because less data is available or because dictionary information is limited or absent. For knowledge-free clustering, our approach can also be extended through the use of infinite models so that the number of clusters need not be specified in advance. We hope that our success with POS tagging will inspire further research into Bayesian methods for other natural language learning tasks.

References

M. Banko and R. Moore. 2004. A study of unsupervised part-of-speech tagging. In *Proceedings of COLING '04*.

E. Brill. 1995. Unsupervised learning of disambiguation rules for part of speech tagging. In *Proceedings of the 3rd Workshop on Very Large Corpora*, pages 1–13.

P. Brown, V. Della Pietra, V. de Souza, J. Lai, and R. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479.

A. Clark. 2000. Inducing syntactic categories by context distribution clustering. In *Proceedings of the Conference on Natural Language Learning (CONLL)*.

S. Finch, N. Chater, and M. Redington. 1995. Acquiring syntactic information from distributional statistics. In J. In Levy, D. Bairaktaris, J. Bullinaria, and P. Cairns, editors, *Connectionist Models of Memory and Language*. UCL Press, London.

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

W.R. Gilks, S. Richardson, and D. J. Spiegelhalter, editors. 1996. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, Suffolk.

A. Haghighi and D. Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of HLT-NAACL*.

M. Johnson, T. Griffiths, and S. Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo.

D. Klein and C. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the ACL*.

D. MacKay and L. Bauman Peto. 1995. A hierarchical Dirichlet language model. *Natural Language Engineering*, 1:289–307.

M. Meilă. 2002. Comparing clusterings. Technical Report 418, University of Washington Statistics Department.

B. Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–172.

L. Saul and F. Pereira. 1997. Aggregate and mixed-order markov models for statistical language processing. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

H. Schütze. 1995. Distributional part-of-speech tagging. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*.

N. Smith and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of ACL*.

I. Wang and D. Schuurmans. 2005. Improved estimation for unsupervised part-of-speech tagging. In *Proceedings of the IEEE International Conference on Natural Language Processing and Knowledge Engineering (IEEE NLP-KE)*.

Computationally Efficient M-Estimation of Log-Linear Structure Models*

Noah A. Smith and **Douglas L. Vail** and **John D. Lafferty**
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA
{nasmith,dvail2,lafferty}@cs.cmu.edu

Abstract

We describe a new loss function, due to Jeon and Lin (2006), for estimating structured log-linear models on arbitrary features. The loss function can be seen as a (generative) alternative to maximum likelihood estimation with an interesting information-theoretic interpretation, and it is statistically consistent. It is substantially faster than maximum (conditional) likelihood estimation of conditional random fields (Lafferty et al., 2001; an order of magnitude or more). We compare its performance and training time to an HMM, a CRF, an MEMM, and pseudolikelihood on a shallow parsing task. These experiments help tease apart the contributions of rich features and discriminative training, which are shown to be more than additive.

1 Introduction

Log-linear models are a very popular tool in natural language processing, and are often lauded for permitting the use of "arbitrary" and "correlated" features of the data by a model. Users of log-linear models know, however, that this claim requires some qualification: any feature is permitted in principle, but training log-linear models (and decoding under them) is tractable only when the model's independence assumptions permit efficient inference procedures. For example, in the original conditional random fields (Lafferty et al., 2001), features were confined to locally-factored indicators on label bigrams and label unigrams (with any of the observation).

Even in cases where inference in log-linear models is tractable, it requires the computation of a partition function. More formally, a log-linear model for random variables X and Y over $\mathcal{X}, \mathcal{Y}$ defines:

$$p_{\mathbf{w}}(x, y) = \frac{e^{\mathbf{w}^\top \mathbf{f}(x,y)}}{\sum_{x',y' \in \mathcal{X} \times \mathcal{Y}} e^{\mathbf{w}^\top \mathbf{f}(x',y')}} = \frac{e^{\mathbf{w}^\top \mathbf{f}(x,y)}}{Z(\mathbf{w})} \quad (1)$$

where $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^m$ is the feature vector-function and $\mathbf{w} \in \mathbb{R}^m$ is a weight vector that parameterizes the model. In NLP, we rarely train this model by maximizing *likelihood*, because the partition function $Z(\mathbf{w})$ is expensive to compute exactly. $Z(\mathbf{w})$ can be approximated (e.g., using Gibbs sampling; Rosenfeld, 1997).

In this paper, we propose the use of a new loss function that is computationally **efficient** and statistically **consistent** (§2). Notably, repeated inference is not required during estimation. This loss function can be seen as a case of M-estimation[1] that was originally developed by Jeon and Lin (2006) for nonparametric density estimation. This paper gives an information-theoretic motivation that helps elucidate the objective function (§3), shows how to apply the new estimator to structured models used in NLP (§4), and compares it to a state-of-the-art noun phrase chunker (§5). We discuss implications and future directions in §6.

2 Loss Function

As before, let X be a random variable over a high-dimensional space $\mathcal{X}$, and similarly Y over $\mathcal{Y}$. $\mathcal{X}$

*This work was supported by NSF grant IIS-0427206 and the DARPA CALO project. The authors are grateful for feedback from David Smith and from three anonymous ACL reviewers, and helpful discussions with Charles Sutton.

[1] "M-estimation" is a generalization of MLE (van der Vaart, 1998); space does not permit a full discussion.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 752–759,
Prague, Czech Republic, June 2007.

might be the set of all sentences in a language, and $\mathcal{Y}$ the set of all POS tag sequences or the set of all parse trees. Let q_0 be a "base" distribution that is our first approximation to the true distribution over $\mathcal{X} \times \mathcal{Y}$. HMMs and PCFGs, while less accurate as predictors than the rich-featured log-linear models we desire, might be used to define q_0.

The model we estimate will have the form

$$p_{\mathbf{w}}(x, y) \propto q_0(x, y) e^{\mathbf{w}^\top \mathbf{f}(x,y)} \tag{2}$$

Notice that $p_{\mathbf{w}}(x, y) = 0$ whenever $q_0(x, y) = 0$. It is therefore important for q_0 to be smooth, since the support of $p_{\mathbf{w}}$ is a subset of the support of q_0. Notice that we have not written the partition function explicitly in Eq. 2; it will never need to be computed during estimation or inference. The unnormalized distribution will suffice for all computation.

Suppose we have observations $\langle x_1, x_2, ..., x_n \rangle$ with annotations $\langle y_1, ..., y_n \rangle$. The (unregularized) loss function, due to Jeon and Lin (2006), is[2]

$$\begin{aligned}
\ell(\mathbf{w}) &= \frac{1}{n} \sum_{i=1}^{n} e^{-\mathbf{w}^\top \mathbf{f}(x_i, y_i)} \\
&\quad + \sum_{x,y} q_0(x, y) \left(\mathbf{w}^\top \mathbf{f}(x, y) \right) \qquad (3) \\
&= \frac{1}{n} \sum_{i=1}^{n} e^{-\mathbf{w}^\top \mathbf{f}(x_i, y_i)} + \mathbf{w}^\top \sum_{x,y} q_0(x, y) \mathbf{f}(x, y) \\
&= \frac{1}{n} \sum_{i=1}^{n} e^{-\mathbf{w}^\top \mathbf{f}(x_i, y_i)} + \mathbf{w}^\top \underbrace{\mathbf{E}_{q_0(X,Y)}[\mathbf{f}(X, Y)]}_{\text{constant}(\mathbf{w})}
\end{aligned}$$

Before explaining this objective, we point out some attractive computational properties. Notice that $\mathbf{f}(x_i, y_i)$ (for all i) and the expectations of the feature vectors under q_0 are *constant* with respect to $\mathbf{w}$. Computing the function in Eq. 3, then, requires no inference and no dynamic programming, only $O(nm)$ floating-point operations.

3 An Interpretation

Here we give an account of the loss function as a way of "cleaning up" a mediocre model (q_0). We show that this estimate aims to *model* a presumed perturbation that created q_0, by minimizing the KL divergence between q_0 and a perturbed version of the sample distribution $\tilde{p}$.

Consider Eq. 2. Given a training dataset, maximizing *likelihood* under this model means assuming that there is some $\mathbf{w}^*$ for which the true distribution $p^*(x, y) = p_{\mathbf{w}^*}(x, y)$. Carrying out MLE, however, would require computing the partition function $\sum_{x',y'} q_0(x', y') e^{\mathbf{w}^\top \mathbf{f}(x',y')}$, which is in general intractable. Rearranging Eq. 2 slightly, we have

$$q_0(x, y) \propto p^*(x, y) e^{-\mathbf{w}^\top \mathbf{f}(x,y)} \tag{4}$$

If q_0 is close to the true model, $e^{-\mathbf{w}^\top \mathbf{f}(x,y)}$ should be close to 1 and $\mathbf{w}$ close to zero. In the sequence model setting, for example, if q_0 is an HMM that explains the data well, then the additional features are not necessary (equivalently, their weights should be 0). If q_0 is imperfect, we might wish to make it more powerful by adding features (e.g., $\mathbf{f}$), but q_0 nonetheless provides a reasonable "starting point" for defining our model.

So instead of maximizing likelihood, we will *minimize* the KL divergence between the two sides of Eq. 4.[3]

$$\begin{aligned}
&D_{\mathrm{KL}}(q_0(x, y) \| p^*(x, y) e^{-\mathbf{w}^\top \mathbf{f}(x,y)}) \qquad (5) \\
&= \sum_{x,y} q_0(x, y) \log \frac{q_0(x, y)}{p^*(x, y) e^{-\mathbf{w}^\top \mathbf{f}(x,y)}} \qquad (6) \\
&\quad + \sum_{x,y} p^*(x, y) e^{-\mathbf{w}^\top \mathbf{f}(x,y)} - \sum_{x,y} q_0(x, y) \\
&= -H(q_0) + \sum_{x,y} p^*(x, y) e^{-\mathbf{w}^\top \mathbf{f}(x,y)} - 1 \\
&\quad - \sum_{x,y} q_0(x, y) \log \left(p^*(x, y) e^{-\mathbf{w}^\top \mathbf{f}(x,y)} \right) \\
&= \text{constant}(\mathbf{w}) + \sum_{x,y} p^*(x, y) e^{-\mathbf{w}^\top \mathbf{f}(x,y)} \\
&\quad + \sum_{x,y} q_0(x, y) \left(\mathbf{w}^\top \mathbf{f}(x, y) \right)
\end{aligned}$$

[2] We give only the discrete version here, because it is most relevant for an ACL audience. Also, our linear function $\mathbf{w}^\top \mathbf{f}(x_i, y_i)$ is a simple case; another kernel (for example) could be used.

[3] The KL divergence here is generalized for unnormalized distributions, following O'Sullivan (1998):

$$D_{\mathrm{KL}}(\mathbf{u} \| \mathbf{v}) = \textstyle\sum_j \left(u_j \log \frac{u_j}{v_j} - u_j + v_j \right)$$

where $\mathbf{u}$ and $\mathbf{v}$ are nonnegative vectors defining unnormalized distributions over the same event space. Note that when $\sum_j u_j = \sum_j v_j = 1$, this formula takes on the more familiar form, as $-\sum_j u_j$ and $\sum_j v_j$ cancel.

If we replace p^* with the empirical (sampled) distribution $\tilde{p}$, minimizing the above KL divergence is equivalent to minimizing $\ell(\mathbf{w})$ (Eq. 3). It may be helpful to think of $-\mathbf{w}$ as the parameters of a process that "damage" the true model p^*, producing q_0, and the estimation of $\mathbf{w}$ as learning to undo that damage.

In the remainder of the paper, we use the general term "M-estimation" to refer to the minimization of $\ell(\mathbf{w})$ as a way of training a log-linear model.

4 Algorithms for Models of Sequences and Trees

We discuss here some implementation aspects of the application of M-estimation to NLP models.

4.1 Expectations under q_0

The base distribution q_0 enters into implementation in two places: $\mathbf{E}_{q_0(X,Y)}[\mathbf{f}(X,Y)]$ must be computed for training, and $q_0(x,y)$ is a factor in the model used in *decoding*.

If q_0 is a familiar stochastic grammar, such as an HMM or a PCFG, or any generative model from which sampling is straightforward, it is possible to estimate the feature expectations by sampling from the model directly; for sample $\langle(\tilde{x}_i, \tilde{y}_i)\rangle_{i=1}^{s}$ let:

$$\mathbf{E}_{q_0(X,Y)}[f_j(X,Y)] \leftarrow \frac{1}{s}\sum_{i=1}^{s} f_j(\tilde{x}_i, \tilde{y}_i) \quad (7)$$

If the feature space is sparse under q_0 (likely in most settings), then smoothing may be required.

If q_0 is an HMM or a PCFG, the expectation vector can be computed exactly by solving a system of equations. We will see that for the common cases where features are local substructures, inference is straightforward. We briefly describe how this can be done for a bigram HMM and a PCFG.

4.1.1 Expectations under an HMM

Let $\mathcal{S}$ be the state space of a first-order HMM. If $\mathbf{s} = \langle s_1, ..., s_k\rangle$ is a state sequence and $\mathbf{x} = \langle x_1, ..., x_k\rangle$ is an observed sequence of emissions, then:

$$q_0(\mathbf{s}, \mathbf{x}) = \left(\prod_{i=1}^{k} t_{s_{i-1}}(s_i) e_{s_i}(x_i)\right) t_{s_k}(\text{stop}) \quad (8)$$

(Assume $s_0 = \text{start}$ is the single, silent, initial state, and stop is the only stop state, also silent. We assume no other states are silent.)

The first step is to compute path-sums *into* and *out of* each state, under the HMM q_0. To do this, define i_s as the total weight of state-prefixes (beginning in start) ending in s and o_s as the total weight of state-suffixes beginning in s (and ending in stop):[4]

$$i_{\text{start}} = o_{\text{stop}} = 1 \quad (9)$$

$$\forall s \in \mathcal{S} \setminus \{\text{start}, \text{stop}\}:$$

$$\begin{aligned} i_s &= \sum_{n=1}^{\infty} \sum_{\langle s_1,...,s_n\rangle \in \mathcal{S}^n} \left(\prod_{i=1}^{n} t_{s_{i-1}}(s_i)\right) t_{s_n}(s) \\ &= \sum_{s' \in \mathcal{S}} i_{s'} t_{s'}(s) \quad (10) \\ o_s &= \sum_{n=1}^{\infty} \sum_{\langle s_1,...,s_n\rangle \in \mathcal{S}^n} t_s(s_1) \left(\prod_{i=2}^{n} t_{s_{i-1}}(s_i)\right) \\ &= \sum_{s' \in \mathcal{S}} t_s(s') o_{s'} \quad (11) \end{aligned}$$

This amounts to two linear systems given the transition probabilities t, where the variables are $i_\bullet$ and $o_\bullet$, respectively. In each system there are $|\mathcal{S}|$ variables and $|\mathcal{S}|$ equations. Once solved, expected counts of transition and emission features under q_0 are straightforward:

$$\begin{aligned} \mathbf{E}_{q_0}[s \overset{\text{transit}}{\rightarrow} s'] &= i_s t_s(s') o_{s'} \\ \mathbf{E}_{q_0}[s \overset{\text{emit}}{\rightarrow} x] &= i_s e_s(x) o_s \end{aligned}$$

Given i and o, $\mathbf{E}_{q_0}$ can be computed for other features in the model in a similar way, provided they correspond to contiguous substructures. For example, a feature f_{627} that counts occurrences of "$S_i = s$ and $X_{i+3} = x$" has expected value $\mathbf{E}_{q_0}[f_{627}] =$

$$\sum_{s',s'',s''' \in \mathcal{S}} i_s t_s(s') t_{s'}(s'') t_{s''}(s''') e_{s'''}(x) o_{s'''} \quad (12)$$

Non-contiguous substructure features with "gaps" require summing over paths between any pair of states. This is straightforward (we omit it for space), but of course using such features (while interesting) would complicate inference in decoding.

[4] It may be helpful to think of i as forward probabilities, but for the observation set $\mathcal{Y}^*$ rather than a *particular* observation $\mathbf{y}$. o are like backward probabilities. Note that, because some counted prefixes are prefixes of others, i can be > 1; similarly for o.

4.1.2 Expectations under a PCFG

In general, the expectations for a PCFG require solving a quadratic system of equations. The analogy this time is to inside and outside probabilities. Let the PCFG have nonterminal set $\mathcal{N}$, start symbol $S \in \mathcal{N}$, terminal alphabet Σ, and rules of the form $A \rightarrow B\ C$ and $A \rightarrow x$. (We assume Chomsky normal form for clarity; the generalization is straightforward.) Let $r_A(B\ C)$ and $r_A(x)$ denote the probabilities of nonterminal A rewriting to child sequence $B\ C$ or x, respectively. Then $\forall A \in \mathcal{N}$:

$$\begin{aligned} o_A &= \sum_{B \in \mathcal{N}} \sum_{C \in \mathcal{N}} o_B i_C [r_B(A\ C) + r_B(C\ A)] \\ &\quad + \begin{cases} 1 & \text{if } A = S \\ 0 & \text{otherwise} \end{cases} \\ i_A &= \sum_{B \in \mathcal{N}} \sum_{C \in \mathcal{N}} r_A(B\ C) i_B i_C\ + \sum_x r_A(x) i_x \\ o_x &= \sum_{A \in \mathcal{N}} o_A r_A(x), \forall x \in \Sigma \\ i_x &= 1, \forall x \in \Sigma \end{aligned}$$

In most practical applications, the PCFG will be "tight" (Booth and Thompson, 1973; Chi and Geman, 1998). Informally, this means that the probability of a derivation rooted in S failing to terminate is zero. If that is the case, then $i_A = 1$ for all $A \in \mathcal{N}$, and the system becomes linear (see also Corazza and Satta, 2006).[5] If tightness is not guaranteed, iterative propagation of weights, following Stolcke (1995), works well in our experience for solving the quadratic system, and converges quickly.

As in the HMM case, expected counts of arbitrary contiguous tree substructures can be computed as products of probabilities of rules appearing within the structure, factoring in the o value of the structure's root and the i values of the structure's leaves.

4.2 Optimization

To carry out M-estimation, we minimize the function $\ell(\mathbf{w})$ in Eq. 3. To apply gradient descent or a quasi-Newton numerical optimization method,[6] it suffices to specify the fixed quantities $\mathbf{f}(x_i, y_i)$ (for all $i \in \{1, 2, ..., n\}$) and the vector $\mathbf{E}_{q_0(X,Y)}[\mathbf{f}(X, Y)]$. The gradient is:[7]

$$\frac{\partial \ell}{\partial w_j} = -\sum_{i=1}^{n} e^{-\mathbf{w}^\top \mathbf{f}(x_i, y_i)} f_j(x_i, y_i) + \mathbf{E}_{q_0}[f_j] \tag{13}$$

The Hessian (matrix of second derivatives) can also be computed with relative ease, though the space requirement could become prohibitive. For problems where m is relatively small, this would allow the use of second-order optimization methods that are likely to converge in fewer iterations.

It is easy to see that Eq. 3 is convex in $\mathbf{w}$. Therefore, convergence to a global optimum is guaranteed and does not depend on the initializing value of $\mathbf{w}$.

4.3 Regularization

Regularization is a technique from pattern recognition that aims to keep parameters (like $\mathbf{w}$) from overfitting the training data. It is crucial to the performance of most statistical learning algorithms, and our experiments show it has a major effect on the success of the M-estimator. Here we use a quadratic regularizer, minimizing $\ell(\mathbf{w}) + (\mathbf{w}^\top \mathbf{w})/2c$. Note that this is also convex and differentiable if $c > 0$. The value of c can be chosen using a tuning dataset. This regularizer aims to keep each coordinate of $\mathbf{w}$ close to zero.

In the M-estimator, regularization is particularly important when the expectation of some feature f_j, $\mathbf{E}_{q_0(X,Y)}[f_j(X, Y)]$ is equal to zero. This can happen either due to sampling error (f_j simply failed to appear with a positive value in the finite sample) or because q_0 assigns zero probability mass to any $x \in \mathcal{X}, y \in \mathcal{Y}$ where $f_j(x, y) \neq 0$. Without regularization, the weight w_j will tend toward $\pm\infty$, but the quadratic penalty term will prevent that undesirable tendency. Just as the addition of a quadratic regularizer to likelihood can be interpreted as a zero-mean Gaussian prior on $\mathbf{w}$ (Chen and Rosenfeld, 2000), it can be so-interpreted here. The regularized objective is analogous to maximum *a posteriori* estimation.

5 Shallow Parsing

We compared M-estimation to a hidden Markov model and other training methods on English noun

[5] The same is true for HMMs: if the probability of nontermination is zero, then for all $s \in \mathcal{S}$, $o_s = 1$.

[6] We use L-BFGS (Liu and Nocedal, 1989) as implemented in the R language's `optim` function.

[7] Taking the limit as $n \rightarrow \infty$ and setting equal to zero, we have the basis for a proof that $\ell(\mathbf{w})$ is statistically consistent.

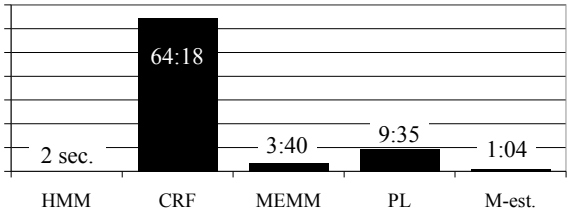


Figure 1: Wall time (hours:minutes) of training the HMM and 100 L-BFGS iterations for each of the extended-feature models on a 2.2 GHz Sun Opteron with 8GB RAM. See discussion in text for details.

phrase (NP) chunking. The dataset comes from the Conference on Natural Language Learning (CoNLL) 2000 shallow parsing shared task (Tjong Kim Sang and Buchholz, 2000); we apply the model to NP chunking only. About 900 sentences were reserved for tuning regularization parameters.

Baseline/q_0 In this experiment, the simple baseline is a second-order HMM. The states correspond to $\{\mathrm{B}, \mathrm{I}, \mathrm{O}\}$ labels, denoting the beginning, inside, and outside of noun phrases. Each state emits a tag and a word (independent of each other given the state). We replaced the first occurrence of every tag and of every word in the training data with an OOV symbol, giving a fixed tag vocabulary of 46 and a fixed word vocabulary of 9,014. Transition distributions were estimated using MLE, and tag- and word-emission distributions were estimated using add-1 smoothing. The HMM had 27,213 parameters. This HMM achieves 86.3% F_1-measure on the development dataset (slightly better than the lowest-scoring of the CoNLL-2000 systems). Heavier or weaker smoothing (an order of magnitude difference in add-λ) of the emission distributions had very little effect. Note that HMM training time is negligible (roughly 2 seconds); it requires counting events, smoothing the counts, and normalizing.

Extended Feature Set Sha and Pereira (2003) applied a conditional random field to the NP chunking task, achieving excellent results. To improve the performance of the HMM and test different estimation methods, we use Sha and Pereira's feature templates, which include subsequences of labels, tags, and words of different lengths and offsets. Here, we use only features observed to occur at least once in the training data, accounting (in addition to our OOV treatment) for the slight drop in performance compared to what Sha and Pereira report. There are 630,862 such features.

	prec.	recall	F_1
HMM features:			
HMM	85.60	88.68	87.11
CRF	90.40	89.56	89.98
PL	80.31	81.37	80.84
MEMM	86.03	88.62	87.31
M-est.	85.57	88.65	87.08
extended features:			
CRF	94.04	93.68	93.86
PL	91.88	91.79	91.83
MEMM	90.89	92.15	91.51
M-est.	88.88	90.42	89.64

Table 1: NP chunking accuracy on test data using different training methods. The effects of discriminative training (CRF) and extended feature sets (lower section) are more than additive.

Using the original HMM feature set and the extended feature set, we trained four models that can use arbitrary features: **conditional random fields** (a near-replication of Sha and Pereira, 2003), **maximum entropy Markov models** (MEMMs; McCallum et al., 2000), **pseudolikelihood** (Besag, 1975; see Toutanova et al., 2003, for a tagging application), and our M-estimator with the HMM as q_0. CRFs and MEMMs are discriminatively-trained to maximize conditional likelihood (the former is parameterized using a sequence-normalized log-linear model, the latter using a locally-normalized log-linear model). Pseudolikelihood is a consistent estimator for the *joint* likelihood, like our M-estimator; its objective function is a sum of log probabilities.

In each case, we trained seven models for each feature set with quadratic regularizers $c \in [10^{-1}, 10]$, spaced at equal intervals in the log-scale, plus an unregularized model ($c = \infty$). As discussed in §4.2, we trained using L-BFGS; training continued until relative improvement fell within machine precision or 100 iterations, whichever came first. After training, the value of c is chosen that maximizes F_1 accuracy on the tuning set.

Runtime Fig. 1 compares the wall time of carefully-timed training runs on a dedicated server. Note that Dyna, a high-level programming language, was used for dynamic programming (in the CRF)

and summations (MEMM and pseudolikelihood). The runtime overhead incurred by using Dyna is estimated as a slow-down factor of 3–5 against a hand-tuned implementation (Eisner et al., 2005), though the slow-down factor is almost certainly less for the MEMM and pseudolikelihood. All training (except the HMM, of course) was done using the R language implementation of L-BFGS. In our implementation, the M-estimator trained substantially faster than the other methods. Of the 64 minutes required to train the M-estimator, 6 minutes were spent precomputing $\mathbf{E}_{q_0(X,Y)}[\mathbf{f}(X,Y)]$ (this need not be repeated if the regularization settings are altered).

Accuracy Tab. 1 shows how NP chunking accuracy compares among the models. With HMM features, the M-estimator is about the same as the HMM and MEMM (better than PL and worse than the CRF). With extended features, the M-estimator lags behind the slower methods, but performs about the same as the HMM-featured CRF (2.5–3 points over the HMM). The full-featured CRF improves performance by another 4 points. Performance as a function of training set size is plotted in Fig. 2; the different methods behave relatively similarly as the training data are reduced. Fig. 3 plots accuracy (on tuning data) against training time, for a variety of training dataset sizes and regularizaton settings, under different training methods. This illustrates the training-time/accuracy tradeoff: the M-estimator, when well-regularized, is considerably faster than the other methods, at the expense of accuracy. This experiment gives some insight into the relative importance of extended **features** versus **estimation methods**. The M-estimated model is, like the maximum likelihood-estimated HMM, a generative model. Unlike the HMM, it uses a much larger set of features–the same features that the discriminative models use. Our result supports the claim that good features are necessary for state-of-the-art performance, but so is good training.

5.1 Effect of the Base Distribution

We now turn to the question of the base distribution q_0: how accurate does it need to be? Given that the M-estimator is consistent, it should be clear that, in the limit and assuming that our model family p is correct, q_0 should not matter (except in its support).

q_0	selection	prec.	recall	F_1
HMM	F_1, prec.	88.88	90.42	89.64
l.u.	F_1	72.91	57.56	64.33
	prec.	84.40	37.68	52.10
emp.	F_1	84.38	89.43	86.83

Table 2: NP chunking accuracy on test data using different base models for the M-estimator. The "selection" column shows which accuracy measure was optimized when selecting the hyperparameter c.

In NLP, we deal with finite datasets and imperfect models, so q_0 may have practical importance.

We next consider an alternative q_0 that is far less powerful; in fact, it is uninformative about the variable to be predicted. Let $\mathbf{x}$ be a sequence of words, $\mathbf{t}$ be a sequence of part-of-speech tags, and $\mathbf{y}$ be a sequence of $\{\mathrm{B}, \mathrm{I}, \mathrm{O}\}$-labels. The model is:

$$q_0^{\text{l.u.}}(\mathbf{x}, \mathbf{t}, \mathbf{y}) \stackrel{\text{def}}{=} \left(\prod_{i=1}^{|\mathbf{x}|} p_{\text{uni}}(x_i) p_{\text{uni}}(t_i) \frac{1}{N_{y_{i-1}}} \right) \frac{1}{N_{y_{|\mathbf{x}|}}} \tag{14}$$

where N_y is the number of labels (including stop) that can follow y (3 for O and $y_0 = \text{start}$, 4 for B and I). p_{uni} are the tag and word unigram distributions, estimated using MLE with add-1 smoothing. This model ignores temporal effects. On its own, this model achieves 0% precision and recall, because it labels every word O (the most likely label sequence is $\mathrm{O}^{|\mathbf{x}|}$). We call this model **l.u.** ("locally uniform").

Tab. 2 shows that, while an M-estimate that uses $q_0^{\text{l.u.}}$ is not nearly as accurate as the one based on an HMM, the M-estimator *did* manage to improve considerably over $q_0^{\text{l.u.}}$. So the M-estimator is far better than nothing, and in this case, tuning c to maximize precision (rather than F_1) led to an M-estimated model with precision competitive with the HMM. We point this out because, in applications involving very large corpora, a model with good precision may be useful even if its coverage is mediocre.

Another question about q_0 is whether it should take into account all possible values of the input variables (here, $\mathbf{x}$ and $\mathbf{t}$), or only those seen in training. Consider the following model:

$$q_0^{\text{emp}}(\mathbf{x}, \mathbf{t}, \mathbf{y}) \stackrel{\text{def}}{=} q_0(\mathbf{y} \mid \mathbf{x}, \mathbf{t}) \tilde{p}(\mathbf{x}, \mathbf{t}) \tag{15}$$

Here we use the empirical distribution over tag/word

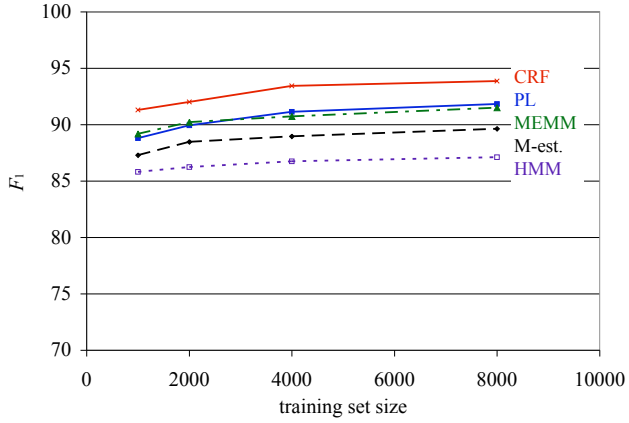


Figure 2: Learning curves for different estimators; all of these estimators except the HMM use the extended feature set.

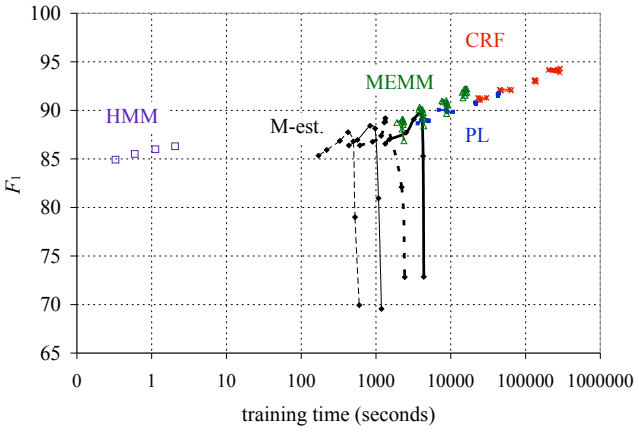


Figure 3: Accuracy (tuning data) vs. training time. The M-estimator trains notably faster. The points in a given curve correspond to different regularization strengths (c); M-estimation is more damaged by weak than strong regularization.

sequences, and the HMM to define the distribution over label sequences. The expectations $\mathbf{E}_{q_0^{\text{emp}}(X)}[\mathbf{f}(X)]$ can be computed using dynamic programming over the training data (recall that this only needs to be done once, cf. the CRF). Strictly speaking, q_0^{emp} assigns probability zero to any sequence not seen in training, but we can ignore the $\tilde{p}$ marginal at decoding time. As shown in Tab. 2, this model slightly improves recall over the HMM, but damages precision; the gains of M-estimation seen with the HMM as q_0, are not reproduced. From these experiments, we conclude that the M-estimator might perform considerably better, given a better q_0.

5.2 Input-Only Features

We present briefly one negative result. Noting that the M-estimator is a modeling technique that estimates a distribution over both input and output variables (i.e., a generative model), we wanted a way to make the objective more discriminative while still maintaining the computational property that inference (of any kind) not be required during the inner loop of iterative training.

The idea is to reduce the predictive burden on the feature weights for $\mathbf{f}$. When designing a CRF, features that do not depend on the output variable (here, $\mathbf{y}$) are unnecessary. They cannot distinguish between competing labelings for an input, and so their weights will be set to zero during conditional estimation. The feature vector function in Sha and Pereira's chunking model does not include such features. In M-estimation, however, adding such "input-only" features might permit better modeling of the data and, more importantly, use the original features primarily for the discriminative task of modeling $\mathbf{y}$ *given* the input.

Adding unigram, bigram, and trigram features to $\mathbf{f}$ for M-estimation resulted in a very small decrease in performance: selecting for F_1, this model achieves 89.33 F_1 on test data.

6 Discussion

M-estimation fills a gap in the plethora of training techniques that are available for NLP models today: it permits **arbitrary features** (like so-called conditional "maximum entropy" models such as CRFs) but estimates a **generative** model (permitting, among other things, classification on input variables and meaningful combination with other models). It is similar in spirit to **pseudolikelihood** (Besag, 1975), to which it compares favorably on training runtime and unfavorably on accuracy.

Further, since no inference is required during training, *any* features really are permitted, so long as their expected values can be estimated under the base model q_0. Indeed, M-estimation is considerably easier to implement than conditional estimation. Both require feature counts from the training data; M-estimation replaces repeated calculation and differentiation of normalizing constants with inference or sampling (once) under a base model. So

the M-estimator is much faster to train.

Generative and discriminative models have been compared and discussed a great deal (Ng and Jordan, 2002), including for NLP models (Johnson, 2001; Klein and Manning, 2002). Sutton and McCallum (2005) present approximate methods that keep a discriminative objective while avoiding full inference.

We see M-estimation as a particularly promising method in settings where performance depends on high-dimensional, highly-correlated feature spaces, where the desired features "large," making discriminative training too time-consuming—a compelling example is machine translation. Further, in some settings a locally-normalized conditional log-linear model (like an MEMM) may be difficult to design; our estimator avoids normalization altogether.[8] The M-estimator may also be useful as a tool in designing and selecting feature combinations, since more trials can be run in less time. After selecting a feature set under M-estimation, discriminative training can be applied on that set. The M-estimator might also serve as an *initializer* to discriminative models, perhaps reducing the number of times inference must be performed—this could be particularly useful in very-large data scenarios. In future work we hope to explore the use of the M-estimator within hidden variable learning, such as the Expectation-Maximization algorithm (Dempster et al., 1977).

7 Conclusions

We have presented a new loss function for generatively estimating the parameters of log-linear models. The M-estimator is fast to train, requiring no repeated, expensive calculation of normalization terms. It was shown to improve performance on a shallow parsing task over a baseline (generative) HMM, but it is not competitive with the state-of-the-art. Our sequence modeling experiments support the widely accepted claim that discriminative, rich-feature modeling works as well as it does *not just* because of rich features in the model, but also because of discriminative training. Our technique fills an important gap in the spectrum of learning methods for NLP models and shows promise for application when discriminative methods are too expensive.

[8] Note that MEMMs also require local partition functions—which may be expensive—to be computed at decoding time.

References

J. E. Besag. 1975. Statistical analysis of non-lattice data. *The Statistician*, 24:179–195.

T. L. Booth and R. A. Thompson. 1973. Applying probability measures to abstract languages. *IEEE Transactions on Computers*, 22(5):442–450.

S. Chen and R. Rosenfeld. 2000. A survey of smoothing techniques for ME models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50.

Z. Chi and S. Geman. 1998. Estimation of probabilistic context-free grammars. *Computational Linguistics*, 24(2):299–305.

A. Corazza and G. Satta. 2006. Cross-entropy and estimation of probabilistic context-free grammars. In *Proc. of HLT-NAACL*.

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

J. Eisner, E. Goldlust, and N. A. Smith. 2005. Compiling Comp Ling: Practical weighted dynamic programming and the Dyna language. In *Proc. of HLT-EMNLP*.

Y. Jeon and Y. Lin. 2006. An effective method for high-dimensional log-density ANOVA estimation, with application to nonparametric graphical model building. *Statistical Sinica*, 16:353–374.

M. Johnson. 2001. Joint and conditional estimation of tagging and parsing models. In *Proc. of ACL*.

D. Klein and C. D. Manning. 2002. Conditional structure vs. conditional estimation in NLP models. In *Proc. of EMNLP*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*.

D. C. Liu and J. Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45:503–528.

A. McCallum, D. Freitag, and F. Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proc. of ICML*.

A. Ng and M. Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naïve Bayes. In *NIPS 14*.

J. A. O'Sullivan. 1998. Alternating minimization algorithms: from Blahut-Armijo to Expectation-Maximization. In A. Vardy, editor, *Codes, Curves, and Signals: Common Threads in Communications*, pages 173–192. Kluwer.

R. Rosenfeld. 1997. A whole sentence maximum entropy language model. In *Proc. of ASRU*.

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proc. of HLT-NAACL*.

A. Stolcke. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201.

C. Sutton and A. McCallum. 2005. Piecewise training of undirected models. In *Proc. of UAI*.

E. F. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proc. of CoNLL*.

K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. of HLT-NAACL*.

A. W. van der Vaart. 1998. *Asymptotic Statistics*. Cambridge University Press.

Guided Learning for Bidirectional Sequence Classification

Libin Shen
BBN Technologies
Cambridge, MA 02138, USA
lshen@bbn.com

Giorgio Satta
Dept. of Inf. Eng'g.
University of Padua
I-35131 Padova, Italy
satta@dei.unipd.it

Aravind K. Joshi
Department of CIS
University of Pennsylvania
Philadelphia, PA 19104, USA
joshi@seas.upenn.edu

Abstract

In this paper, we propose guided learning, a new learning framework for bidirectional sequence classification. The tasks of learning the order of inference and training the local classifier are dynamically incorporated into a single Perceptron like learning algorithm. We apply this novel learning algorithm to POS tagging. It obtains an error rate of 2.67% on the standard PTB test set, which represents 3.3% relative error reduction over the previous best result on the same data set, while using fewer features.

1 Introduction

Many NLP tasks can be modeled as a sequence classification problem, such as POS tagging, chunking, and incremental parsing. A traditional method to solve this problem is to decompose the whole task into a set of individual tasks for each token in the input sequence, and solve these small tasks in a fixed order, usually from left to right. In this way, the output of the previous small tasks can be used as the input of the later tasks. HMM and MaxEnt Markov Model are examples of this method.

Lafferty et al. (2001) showed that this approach suffered from the so called *label bias problem* (Bottou, 1991). They proposed Conditional Random Fields (CRF) as a general solution for sequence classification. CRF models a sequence as an undirected graph, which means that all the individual tasks are solved simultaneously. Taskar et al. (2003) improved the CRF method by employing the large margin method to separate the gold standard sequence labeling from incorrect labellings. However, the complexity of quadratic programming for the large margin approach prevented it from being used in large scale NLP tasks.

Collins (2002) proposed a Perceptron like learning algorithm to solve sequence classification in the traditional left-to-right order. This solution does not suffer from the label bias problem. Compared to the undirected methods, the Perceptron like algorithm is faster in training. In this paper, we will improve upon Collins' algorithm by introducing a bidirectional searching strategy, so as to effectively utilize more context information at little extra cost.

When a bidirectional strategy is used, the main problem is how to select the order of inference. Tsuruoka and Tsujii (2005) proposed the *easiest-first* approach which greatly reduced the computation complexity of inference while maintaining the accuracy on labeling. However, the easiest-first approach only serves as a heuristic rule. The order of inference is not incorporated into the training of the MaxEnt classifier for individual labeling.

Here, we will propose a novel learning framework, namely *guided learning*, to integrate classification of individual tokens and inference order selection into a single learning task. We proposed a Perceptron like learning algorithm (Collins and Roark, 2004; Daumé III and Marcu, 2005) for guided learning. We apply this algorithm to POS tagging, a classic sequence learning problem. Our system reports an error rate of 2.67% on the standard PTB test set, a relative 3.3% error reduction of the previous best system (Toutanova et al., 2003) by using fewer features. By using deterministic search, it obtains an error rate of 2.73%, a 5.9% relative error reduction

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 760–767,
Prague, Czech Republic, June 2007.

over the previous best deterministic algorithm (Tsuruoka and Tsujii, 2005).

The new POS tagger is similar to (Toutanova et al., 2003; Tsuruoka and Tsujii, 2005) in the way that we employ context features. We use a bidirectional search strategy (Woods, 1976; Satta and Stock, 1994), and our algorithm is based on Perceptron learning (Collins, 2002). A unique contribution of our work is on the integration of individual classification and inference order selection, which are learned simultaneously.

2 Guided Learning for Bidirectional Labeling

We first present an example of POS tagging to show the idea of bidirectional labeling. Then we present the inference algorithm and the learning algorithm.

2.1 An Example of POS tagging

Suppose that we have an input sentence

```
Agatha found that book interesting
  w1    w2    w3   w4      w5
            (Step 0)
```

If we scan from left to right, we may find it difficult to resolve the ambiguity of the label for `that`, which could be either DT (determiner), or IN (preposition or subordinating conjunction) in the Penn Treebank. However, if we resolve the labels for `book` and `interesting`, it would be relatively easy to figure out the correct label for `that`.

Now, we show how bidirectional inference works on this sample. Suppose we use beam search with width of 2, and we use a window of (-2, 2) for context features.

For the first step, we enumerate hypotheses for each word. For example, `found` could have a label VBN or VBD. Suppose that at this point the most favorable action, out of the candidate hypotheses, is the assignment of NN to `book`, according to the context features defined on words. Then, we resolve the label for `book` first. We maintain the top two hypotheses as shown below. Here, the second most favorable label for `book` is VB.

```
                  NN
                  VB
Agatha found that book interesting
  w1    w2    w3   w4      w5
            (Step 1)
```

At the second step, assume the most favorable action is the assignment of label JJ to `interesting` in the context of NN for `book`. Then we maintain the top two hypotheses for span `book interesting` as shown below. The second most favorable label for `interesting` is still JJ, but in the context of VB for `book`.

```
                  NN------JJ
                  VB------JJ
Agatha found that book interesting
  w1    w2    w3   w4      w5
            (Step 2)
```

Then, suppose we are most confident for assigning labels VBD and VBN to `found`, in that order. We get two separated tagged spans as shown below.

```
        VBD         NN------JJ
        VBN         VB------JJ
Agatha found that book interesting
  w1    w2    w3   w4      w5
            (Step 3)
```

In the next step, suppose we are most confident for assigning label DT to `that` under the context of VBD on the left and NN-JJ on the right side, as shown below (second most favorable action, not discussed here, is also displayed). After tagging w_3, two separated spans merge into one, starting from `found` to `interesting`.

```
        VBD---DT---NN------JJ
        VBD---IN---NN------JJ
Agatha found that book interesting
  w1    w2    w3   w4      w5
            (Step 4)
```

For the last step, we assign label NNP to `Agatha`, which could be an out-of-vocabulary word, under the context of VBD-DT on the right.

```
  NNP---VBD---DT---NN------JJ
  NNP---VBD---IN---NN------JJ
Agatha found that book interesting
  w1    w2    w3   w4      w5
            (Step 5)
```

This simple example has shown the advantage of adopting a flexible search strategy. However, it is still unclear how we maintain the hypotheses, how we keep candidates and accepted labels and spans, and how we employ dynamic programming. We will answer these questions in the formal definition of the inference algorithm in the next section.

2.2 Inference Algorithm

Terminology: Let the input sequence be $w_1 w_2 \cdots w_n$. For each token w_i, we are expected to assign a label $t_i \in \mathbf{T}$, with $\mathbf{T}$ the label set.

A subsequence $w_i \cdots w_j$ is called a **span**, and is denoted $[i, j]$. Each span p considered by the algorithm is associated with one or more **hypotheses**, that is, sequences over $\mathbf{T}$ having the same length as p. Part of the label sequence of each hypothesis is used as a context for labeling tokens outside the span p. For example, if a tri-gram model is adopted, we use the two labels on the left boundary and the two labels on the right boundary of the hypothesis for labeling outside tokens. The left two labels are called the **left interface**, and the right two labels are called the **right interface**. Left and right interfaces have only one label in case of spans of length one.

A pair $s = (I_{left}, I_{right})$ with a left and a right interface is called a **state**. We partition the hypotheses associated with span p into sets compatible with the same state. In practice, for span p, we use a matrix M_p indexed by states, so that $M_p(s)$, $s = (I_{left}, I_{right})$, is the set of all hypotheses associated with p that are compatible with I_{left} and I_{right}.

For a span p and a state s, we denote the associated top hypothesis as

$$s.T = \operatorname*{argmax}_{h \in M_p(s)} V(h),$$

where V is the score of a hypothesis (defined in (1) below). Similarly, we denote the top state for p as

$$p.S = \operatorname*{argmax}_{s:\, M_p(s) \neq \emptyset} V(s.T).$$

Therefore, for each span p, we have a top hypothesis $p.S.T$, whose score is the highest among all the hypotheses for span p.

Hypotheses are started and grown by means of labeling actions. For each hypothesis h associated with a span p we maintain its most recent labeling action $h.A$, involving some token within p, as well as the states $h.S_L$ and $h.S_R$ that have been used as context by such an action, if any. Note that $h.S_L$ and $h.S_R$ refer to spans that are subsequences of p. We recursively compute the score of h as

$$V(h) = V(h.S_L.T) + V(h.S_R.T) + U(h.A), \quad (1)$$

Algorithm 1 Inference Algorithm

Require: token sequence $w_1 \cdots w_n$;
Require: beam width B;
Require: weight vector $\mathbf{w}$;
1: Initialize P, the set of accepted spans;
2: Initialize Q, the queue of candidate spans;
3: **repeat**
4: span $p' \leftarrow \operatorname{argmax}_{p \in Q} U(p.S.T.A)$;
5: Update P with p';
6: Update Q with p' and P;
7: **until** ($Q = \emptyset$)

where U is the score of an action. In other words, the score of an hypothesis is the sum of the score of the most recent action $h.A$ and the scores of the top hypotheses of the context states. The score of an action $h.A$ is computed through a linear function whose weight vector is $\mathbf{w}$, as

$$U(h.A) = \mathbf{w} \cdot \mathbf{f}(h.A), \quad (2)$$

where $\mathbf{f}(h.A)$ is the feature vector of action $h.A$, which depends on $h.S_L$ and $h.S_R$.

Algorithm: Algorithm 1 is the inference algorithm. We are given the input sequence and two parameters, beam width B to determine the number of states maintained for each span, and weight vector $\mathbf{w}$ used to compute the score of an action.

We first initialize the set P of accepted spans with the empty set. Then we initialize the queue Q of candidate spans with span $[i, i]$ for each token w_i, and for each $t \in \mathbf{T}$ assigned to w_i we set

$$M_{[i,i]}((t, t)) = \{i \rightarrow t\},$$

where $i \rightarrow t$ represents the hypothesis consisting of a single action which assigns label t to w_i. This provides the set of starting hypotheses.

As for the example `Agatha found that book interesting` in the previous subsection, we have

- $P = \emptyset$
- $Q = \{[1, 1], [2, 2], [3, 3], [4, 4], [5, 5]\}$

Suppose NN and VB are the two possible POS tags for w_4 `book`. We have

- $M_{[4,4]}(\text{NN, NN}) = \{h_{441} = 4 \rightarrow \text{NN}\}$
- $M_{[4,4]}(\text{VB, VB}) = \{h_{442} = 4 \rightarrow \text{VB}\}$

The most recent action of hypothesis h_{441} is to assign NN to w_4. According to Equation (2), the score

of this action $U(h_{441}.A)$ depends on the features defined on the local context of action. For example,

$$f_{1001}(h_{441}.A) = \begin{cases} 1 & \text{if } t = \text{NN} \wedge w^{-1} = \texttt{that} \\ 0 & \text{otherwise,} \end{cases}$$

where w^{-1} represents the left word. It should be noted that, for all the features depending on the neighboring tags, the value is always 0, since those tags are still unknown in the step of initialization. Since this operation does not depend on solved tags, we have $V(h_{441}) = U(h_{411}.A)$, according to Equation (1).

The core of the algorithm repeatedly selects a candidate span from Q, and uses it to update P and Q, until a span covering the whole sequence is added to P and Q becomes empty. This is explained in detail below.

At each step, we remove from Q the span p' such that the *action* (not hypothesis) score of its top hypothesis, $p'.S.T$, is the highest. This represents the labeling action for the next move that we are most confident about. Now we need to update P and Q with the selected span p'. We add p' to P, and remove from P the spans included in p', if any. Let $\mathcal{S}$ be the set of removed spans. We remove from Q each span which takes one of the spans in $\mathcal{S}$ as context, and replace it with a new candidate span taking p' (and another accepted span) as context. We always maintain B different states for each span.

Back to the previous example, after Step 3 is completed, w_2 `found`, w_4 `book` and w_5 `interesting` have been tagged and we have

- $P = \{[2, 2], [4, 5]\}$
- $Q = \{[1, 2], [2, 5]\}$

There are two candidate spans in Q, each with its associated hypotheses and most recent actions. More specifically, we can either solve w_1 based on the context hypotheses for $[2, 2]$, resulting in span $[1, 2]$, or else solve w_3 based on the context hypotheses in $[2, 2]$ and $[4, 5]$, resulting in span $[2, 5]$.

The top two states for span $[2, 2]$ are

- $M_{[2,2]}(\text{VBD, VBD}) = \{h_{221} = 2 \rightarrow \text{VBD}\}$
- $M_{[2,2]}(\text{VBN, VBN}) = \{h_{222} = 2 \rightarrow \text{VBN}\}$

and the top two states for span $[4, 5]$ are

- $M_{[4,5]}(\text{NN-JJ, NN-JJ})$ $= \{h_{451} = (\text{NN,NN})5 \rightarrow \text{JJ}\}$
- $M_{[4,5]}(\text{VB-JJ, VB-JJ})$ $= \{h_{452} = (\text{VB,VB})5 \rightarrow \text{JJ}\}$

Here $(\text{NN,NN})5 \rightarrow \text{JJ}$ represents the hypothesis coming from the action of assigning JJ to w_5 under the left context state of (NN,NN). $(\text{VB,VB})5 \rightarrow \text{JJ}$ has a similar meaning.[1]

We first compute the hypotheses resulting from all possible POS tag assignments to w_3, under all possible state combinations of the neighboring spans $[2, 2]$ and $[4, 5]$. Suppose the highest score action consists in the assignment of DT under the left context state (VBD, VBD) and the right context state (NN-JJ, NN-JJ). We obtain hypothesis $h_{251} = (\text{VBD,VBD})3 \rightarrow \text{DT}(\text{NN-JJ, NN-JJ})$ with

$$\begin{aligned} V(h_{251}) &= V((\text{VBD,VBD}).T) + \\ &\quad V((\text{NN-JJ,NN-JJ}).T) + U(h_{251}.A) \\ &= V(h_{221}) + V(h_{451}) + \mathbf{w} \cdot \mathbf{f}(h_{251}.A) \end{aligned}$$

Here, features for action $h_{251}.A$ may depend on the left tag VBD and right tags NN-JJ, which have been solved before. More details of the feature functions are given in Section 4.2. For example, we can have features like

$$f_{2002}(h_{251}.A) = \begin{cases} 1 & \text{if } t = \text{DT} \wedge t^{+2} = \text{JJ} \\ 0 & \text{otherwise,} \end{cases}$$

We maintain the top two states with the highest hypothesis scores, if the beam width is set to two. We have

- $M_{[2,5]}(\text{VBD-DT, NN-JJ}) = \{h_{251} = (\text{VBD,VBD})3 \rightarrow \text{DT}(\text{NN-JJ,NN-JJ})\}$
- $M_{[2,5]}(\text{VBD-IN, NN-JJ}) = \{h_{252} = (\text{VBD,VBD})3 \rightarrow \text{IN}(\text{NN-JJ,NN-JJ})\}$

Similarly, we compute the top hypotheses and states for span $[1, 2]$. Suppose now the hypothesis with the highest *action* score is h_{251}. Then we update P by adding $[2, 5]$ and removing $[2, 2]$ and $[4, 5]$, which are covered by $[2, 5]$. We also update Q by removing $[2, 5]$ and $[1, 2]$,[2] and add new candidate span $[1, 5]$ resulting in

- $P = \{[2, 5]\}$
- $Q = \{[1, 5]\}$

[1] It should be noted that, in these cases, each state contains only one hypothesis. However, if the span is longer than 4 words, there may exist multiple hypotheses for the same state. For example, hypotheses DT-NN-VBD-DT-JJ and DT-NN-VBN-DT-JJ have the same left interface DT-NN and right interface DT-JJ.

[2] Span $[1, 2]$ depends on $[2, 2]$ and $[2, 2]$ has been removed from P. So it is no longer a valid candidate given the accepted spans in P.

The algorithm is especially designed in such a way that, at each step, some new span is added to P or else some spans already present in P are extended by some token(s). Furthermore, no pair of overlapping spans is ever found in P, and the number of pairs of overlapping spans that may be found in Q is always bounded by a constant. This means that the algorithm performs at most n iterations, and its running time is therefore $\mathcal{O}(B^2 n)$, that is, linear in the length of the input sequence.

2.3 Learning Algorithm

In this section, we propose *guided learning*, a Perceptron like algorithm, to learn the weight vector $\mathbf{w}$, as shown in Algorithm 2. We use $p'.G$ to represent the gold standard hypothesis on span p'.

For each input sequence X_r and the gold standard sequence of labeling Y_r, we first initialize P and Q as in the inference algorithm. Then we select the span for the next move as in Algorithm 1. If $p'.S.T$, the top hypothesis of the selected span p', is compatible with the gold standard, we update P and Q as in Algorithm 1. Otherwise, we update the weight vector in the Perceptron style, by promoting the features of the gold standard action, and demoting the features of the action of the top hypothesis. Then we re-generate the queue Q with P and the updated weight vector $\mathbf{w}$. Specifically, we first remove all the elements in Q, and then generate hypotheses for all the possible spans based on the context spans in P. Hypothesis scores and action scores are calculated with the updated weight vector $\mathbf{w}$.

A special aspect of Algorithm 2 is that we maintain two scores: the score of the action represents the confidence for the next move, and the score of the hypothesis represents the overall quality of a partial result. The selection for the next action directly depends on the score of the action, but not on the score of the hypothesis. On the other hand, the score of the hypothesis is used to maintain top partial results for each span.

We briefly describe the soundness of the Guided Learning Algorithm in terms of two aspects. First, in Algorithm 2 weight update is activated whenever there exists an incorrect state s, the action score of whose top hypothesis $s.T$ is higher than that of any state in each span. We demote this action and promote the gold standard action on the same span.

Algorithm 2 Guided Learning Algorithm

Require: training sequence pairs $\{(X_r, Y_r)\}_{1 \le r \le R}$;
Require: beam width B and iterations I;
1: $\mathbf{w} \leftarrow \mathbf{0}$;
2: **for** ($i \leftarrow 1$; $i \le I$; i++) **do**
3: **for** ($r \leftarrow 1$; $r \le R$; r++) **do**
4: Load sequence X_r and gold labeling Y_r.
5: Initialize P, the set of accepted spans
6: Initialize Q, the queue of candidate spans;
7: **repeat**
8: $p' \leftarrow \text{argmax}_{p \in Q}\, U(p.S.T.A)$;
9: **if** ($p'.S.T = p'.G$) **then**
10: Update P with p';
11: Update Q with p' and P;
12: **else**
13: $promote(\mathbf{w}, \mathbf{f}(p'.G.A))$;
14: $demote(\mathbf{w}, \mathbf{f}(p'.S.T.A))$;
15: Re-generate Q with $\mathbf{w}$ and P;
16: **end if**
17: **until** ($Q = \emptyset$)
18: **end for**
19: **end for**

However, we do not automatically adopt the gold standard action on this span. Instead, in the next step, the top hypothesis of another span might be selected based on the score of action, which means that it becomes the most favorable action according to the updated weights.

As a second aspect, if the action score of a gold standard hypothesis is higher than that of any others, this hypothesis and the corresponding span are guaranteed to be selected at line 8 of Algorithm 2. The reason for this is that the scores of the context hypotheses of a gold standard hypothesis must be no less than those of other hypotheses of the same span. This could be shown recursively with respect to Equation 1, because the context hypotheses of a gold standard hypothesis are also compatible with the gold standard.

Furthermore, if we take

$$(\mathbf{x_i} = \mathbf{f}(p'.G.A) - \mathbf{f}(p'.S.T.A), y_i = +1)$$

as a positive sample, and

$$(\mathbf{x_j} = \mathbf{f}(p'.S.T.A) - \mathbf{f}(p'.G.A), y_j = -1)$$

as a negative sample, the weight updates at lines 13

and 14 are a stochastic approximation of gradient descent that minimizes the squared errors of the misclassified samples (Widrow and Hoff, 1960). What is special with our learning algorithm is the strategy used to select samples for training.

In general, this novel learning framework lies between supervised learning and reinforcement learning. Guided learning is more difficult than supervised learning, because we do not know the order of inference. The order is learned automatically, and partial output is in turn used to train the local classifier. Therefore, the order of inference and the local classification are dynamically incorporated in the learning phase.

Guided learning is not as hard as reinforcement learning. At each local step in learning, we always know the undesirable labeling actions according to the gold standard, although we do not know which is the most desirable. In this approach, we can easily collect the automatically generated negative samples, and use them in learning. These negative samples are exactly those we will face during inference with the current weight vector.

In our experiments, we have used Averaged Perceptron (Collins, 2002; Freund and Schapire, 1999) and Perceptron with margin (Krauth and Mézard, 1987) to improve performance.

3 Related Works

Tsuruoka and Tsujii (2005) proposed a bidirectional POS tagger, in which the order of inference is handled with the easiest-first heuristic. Giménez and Màrquez (2004) combined the results of a left-to-right scan and a right-to-left scan. In our model, the order of inference is dynamically incorporated into the training of the local classifier.

Toutanova et al. (2003) reported a POS tagger based on cyclic dependency network. In their work, the order of inference is fixed as from left to right. In this approach, large beam width is required to maintain the ambiguous hypotheses. In our approach, we can handle tokens that we are most confident about first, so that our system does not need a large beam. As shown in Section 4.2, even deterministic inference shows rather good results.

Our guided learning can be modeled as a search algorithm with Perceptron like learning (Daumé III and Marcu, 2005). However, as far as we know, the mechanism of bidirectional search with an online learning algorithm has not been investigated before. In (Daumé III and Marcu, 2005), as well as other similar works (Collins, 2002; Collins and Roark, 2004; Shen and Joshi, 2005), only left-to-right search was employed. Our guided learning algorithm provides more flexibility in search with an automatically learned order. In addition, our treatment of the score of action and the score of hypothesis is unique (see discussion in Section 2.3).

Data Set	Sections	Sentences	Tokens
Training	0-18	38,219	912,344
Develop	19-21	5,527	131,768
Test	22-24	5,462	129,654

Table 1: Data set splits

Furthermore, compared to the above works, our guided learning algorithm is more aggressive on learning. In (Collins and Roark, 2004; Shen and Joshi, 2005), a search stops if there is no hypothesis compatible with the gold standard in the queue of candidates. In (Daumé III and Marcu, 2005), the search is resumed after some gold standard compatible hypotheses are inserted into a queue for future expansion, and the weights are updated correspondingly. However, there is no guarantee that the updated weights assign a higher score to those inserted gold standard compatible hypotheses. In our algorithm, the gold standard compatible hypotheses are used for weight update only. As a result, after each sentence is processed, the weight vector can usually successfully predict the gold standard parse. Therefore our learning algorithm is *aggressive* on weight update.

As far as this aspect is concerned, our algorithm is similar to the MIRA algorithm in (Crammer and Singer, 2003). In MIRA, one always knows the correct hypothesis. In our case, we do not know the correct order of operations. So we use our form of weight update to implement aggressive learning.

4 Experiments on POS Tagging

4.1 Settings

We apply our guided learning algorithm to POS tagging. We carry out experiments on the standard data set of the Penn Treebank (PTB) (Marcus et al., 1994). Following (Ratnaparkhi, 1996; Collins, 2002; Toutanova et al., 2003; Tsuruoka and Tsujii, 2005),

Feature Sets	Templates	Error%
A	Ratnaparkhi's	3.05
B	A + $[t_0, t_1], [t_0, t_{-1}, t_1], [t_0, t_1, t_2]$	2.92
C	B + $[t_0, t_{-2}], [t_0, t_2], [t_0, t_{-2}, w_0], [t_0, t_{-1}, w_0], [t_0, t_1, w_0]$, $[t_0, t_2, w_0]$, $[t_0, t_{-2}, t_{-1}, w_0], [t_0, t_{-1}, t_1, w_0], [t_0, t_1, t_2, w_0]$	2.84
D	C + $[t_0, w_{-1}, w_0], [t_0, w_1, w_0]$	2.78
E	D + $[t_0, X =$ prefix or suffix of $w_0], 4 < \|X\| \leq 9$	2.72

Table 2: Experiments on the *development* data with beam width of 3

we cut the PTB into the training, development and test sets as shown in Table 1. We use tools provided by CoNLL-2005 [3] to extract POS tags from the *mrg* files of PTB. So the data set is the same as previous work. We use the development set to select features and estimate the number of iterations in training. In our experiments, we enumerate all the POS tags for each word instead of using a dictionary as in (Ratnaparkhi, 1996), since the size of the tag set is tractable and our learning algorithm is efficient enough.

4.2 Results

Effect of Features: We first run the experiments to evaluate the effect of features. We use templates to define features. For this set of experiments, we set the beam width $B = 3$ as a balance between speed and accuracy. The guided learning algorithm usually converges on the development data set in 4-8 iterations over the training data.

Table 2 shows the error rate on the development set with different features. We first use the same feature set used in (Ratnaparkhi, 1996), which includes a set of prefix, suffix and lexical features, as well as some bi-gram and tri-gram context features. Following (Collins, 2002), we do not distinguish rare words. On set A, Ratnaparkhi's feature set, our system reports an error rate of 3.05% on the development data set.

With set B, we include a few feature templates which are symmetric to those in Ratnaparkhi's set, but are only available with bidirectional search. With set C, we add more bi-gram and tri-gram features. With set D, we include bi-lexical features. With set E, we use prefixes and suffixes of length up to 9, as in (Toutanova et al., 2003; Tsuruoka and Tsujii, 2005). We obtain 2.72% of error rate. We will use this feature set on our final experiments on the test data.

Effect of Search and Learning Strategies: For the second set of experiments, we evaluate the effect of search methods, learning strategies, and beam width. We use feature set E for this set of experiments. Table 3 shows the error rates on the development data set with both left-to-right (L-to-R) and bidirectional (Bi-Dir) search methods. We also tested both aggressive learning and non-aggressive learning strategies with beam width of 1 and 3.

Search	Aggressive?	Beam=1	Beam=3
L-to-R	Yes	2.94	2.82
L-to-R	No	3.24	2.75
Bi-Dir	Yes	2.84	2.72
Bi-Dir	No	does not converge	

Table 3: Experiments on the *development* data

First, with non-aggressive learning on bidirectional search, the error rate does not converge to a comparable number. This is due to the fact that the search space is too large in bidirectional search, if we do not use aggressive learning to constrain the samples for learning.

With aggressive learning, the bidirectional approach always shows advantages over left-to-right search. However, the gap is not large. This is due to the fact that the accuracy of POS tagging is very high. As a result, we can always keep the gold-standard tags in the beam even with left-to-right search in training.

This can also explain why the performance of left-to-right search with non-aggressive learning is close to bidirectional search if the beam is large enough. However, with beam width = 1, non-aggressive learning over left-to-right search performs much worse, because in this case it is more likely that the gold-standard tag is not in the beam.

This set of experiments show that guided learning is more preferable for tasks with higher ambiguities. In our recent work (Shen and Joshi, 2007), we have applied a variant of this algorithm to dependency parsing, and showed significant improvement over left-to-right non-aggressive learning strategy.

Comparison: Table 4 shows the comparison with the previous works on the PTB test sections.

[3] http://www.lsi.upc.es/~srlconll/soft.html, package srlconll-1.1.tgz.

System	Beam	Error%
(Ratnaparkhi, 1996)	5	3.37
(Tsuruoka and Tsujii, 2005)	1	2.90
(Collins, 2002)	-	2.89
Guided Learning, feature B	3	2.85
(Tsuruoka and Tsujii, 2005)	all	2.85
(Giménez and Màrquez, 2004)	-	2.84
(Toutanova et al., 2003)	-	2.76
Guided Learning, feature E	1	2.73
Guided Learning, feature E	3	2.67

Table 4: Comparison with the previous works

According to the experiments shown above, we build our best system by using feature set E with beam width $B = 3$. The number of iterations on the training data is estimated with respect to the development data. We obtain an error rate of **2.67**% on the test data. With deterministic search, or beam with $B = 1$, we obtain an error rate of 2.73%.

Compared to previous best result on the same data set, 2.76% by (Toutanova et al., 2003), our best result shows a relative error reduction of 3.3%. This result is very promising, since we have not used any specially designed features in our experiments. It is reported in (Toutanova et al., 2003) that a crude company name detector was used to generate features, and it gave rise to significant improvement in performance. However, it is difficult for us to duplicate exactly the same feature for the purpose of comparison, although it is convenient to use features like that in our framework.

5 Conclusions

In this paper, we propose *guided learning*, a new learning framework for bidirectional sequence classification. The tasks of learning the order of inference and training the local classifier are dynamically incorporated into a single Perceptron like algorithm.

We apply this novel algorithm to POS tagging. It obtains an error rate of 2.67% on the standard PTB test set, which represents 3.3% relative error reduction over the previous best result (Toutanova et al., 2003) on the same data set, while using fewer features. By using deterministic search, it obtains an error rate of 2.73%, a 5.9% relative error reduction over the previous best deterministic algorithm (Tsuruoka and Tsujii, 2005). It should be noted that the error rate is close to the inter-annotator discrepancy on PTB, the standard test set for POS tagging, therefore it is very difficult to achieve improvement.

References

L. Bottou. 1991. *Une approche théorique de l'apprentissage connexionniste: Applications à la reconnaissance de la parole.* Ph.D. thesis, Université de Paris XI.

M. Collins and B. Roark. 2004. Incremental parsing with the perceptron algorithm. In *ACL-2004*.

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

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

H. Daumé III and D. Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *ICML-2005*.

Y. Freund and R. E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

J. Giménez and L. Màrquez. 2004. Svmtool: A general pos tagger generator based on support vector machines. In *LREC-2004*.

W. Krauth and M. Mézard. 1987. Learning algorithms with optimal stability in neural networks. *Journal of Physics A*, 20:745–752.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmentation and labeling sequence data. In *ICML-2001*.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

A. Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *EMNLP-1996*.

G. Satta and O. Stock. 1994. Bi-Directional Context-Free Grammar Parsing for Natural Language Processing. *Artificial Intelligence*, 69(1-2).

L. Shen and A. K. Joshi. 2005. Incremental LTAG Parsing. In *EMNLP-2005*.

L. Shen and A. K. Joshi. 2007. Bidirectional LTAG Dependency Parsing. Technical Report 07-02, IRCS, UPenn.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin markov networks. In *NIPS-2003*.

K. Toutanova, D. Klein, C. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL-2003*.

Y. Tsuruoka and J. Tsujii. 2005. Bidirectional inference with the easiest-first strategy for tagging sequence data. In *EMNLP-2005*.

B. Widrow and M. E. Hoff. 1960. Adaptive switching circuits. *IRE WESCON Convention Record, part 4*.

W. Woods. 1976. Parsers in speech understanding systems. Technical Report 3438, Vol. 4, 1–21, BBN Inc.

Different Structures for Evaluating Answers to Complex Questions: Pyramids Won't Topple, and Neither Will Human Assessors

Hoa Trang Dang
Information Access Division
National Institute of Standards and Technology
Gaithersburg, MD 20899
hoa.dang@nist.gov

Jimmy Lin
College of Information Studies
University of Maryland
College Park, MD 20742
jimmylin@umd.edu

Abstract

The idea of "nugget pyramids" has recently been introduced as a refinement to the nugget-based methodology used to evaluate answers to complex questions in the TREC QA tracks. This paper examines data from the 2006 evaluation, the first large-scale deployment of the nugget pyramids scheme. We show that this method of combining judgments of nugget importance from multiple assessors increases the stability and discriminative power of the evaluation while introducing only a small additional burden in terms of manual assessment. We also consider an alternative method for combining assessor opinions, which yields a distinction similar to micro- and macro-averaging in the context of classification tasks. While the two approaches differ in terms of underlying assumptions, their results are nevertheless highly correlated.

1 Introduction

The emergence of question answering (QA) systems for addressing complex information needs has necessitated the development and refinement of new methodologies for evaluating and comparing systems. In the Text REtrieval Conference (TREC) QA tracks organized by the U.S. National Institute of Standards and Technology (NIST), improvements in evaluation processes have kept pace with the evolution of QA tasks. For the past several years, NIST has implemented an evaluation methodology based on the notion of "information nuggets" to assess answers to complex questions. As it has become the *de facto* standard for evaluating such systems, the research community stands to benefit from a better understanding of the characteristics of this evaluation methodology.

This paper explores recent refinements to the nugget-based evaluation methodology developed by NIST. In particular, we examine the recent so-called "pyramid extension" that incorporates relevance judgments from multiple assessors to improve evaluation stability (Lin and Demner-Fushman, 2006).

We organize our discussion as follows: The next section begins by providing a brief overview of nugget-based evaluations and the pyramid extension. Section 3 presents results from the first large-scale implementation of nugget pyramids for QA evaluation in TREC 2006. Analysis shows that this extension improves both stability and discriminative power. In Section 4, we discuss an alternative for combining multiple judgments that parallels the distinction between micro- and macro-averaging often seen in classification tasks. Experiments reveal that the methods yield almost exactly the same results, despite operating on different granularities (individual nuggets vs. individual users).

2 Evaluating Complex Questions

Complex questions are distinguished from factoid questions such as "Who shot Abraham Lincoln?" in that they cannot be answered by named entities (e.g., persons, organizations, dates, etc.). Typically, these information needs are embedded in the context of a scenario (i.e., user task) and often require systems to

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 768–775,
Prague, Czech Republic, June 2007.

synthesize information from multiple documents or to generate answers that cannot be easily extracted (e.g., by leveraging inference capabilities).

To date, NIST has already conducted several large-scale evaluations of complex questions: definition questions in TREC 2003, "Other" questions in TREC 2004–2006, "relationship" questions in TREC 2005, and the complex, interactive QA (ciQA) task in TREC 2006. Definition and Other questions are similar in that they both request novel facts about "targets", which can be persons, organizations, things, and events. Relationship questions evolved into the ciQA task and focus on information needs such as "What financial relationships exist between South American drug cartels and banks in Liechtenstein?" Such complex questions focus on ties (financial, military, familial, etc.) that connect two or more entities. All of these evaluations have employed the nugget-based methodology, which demonstrates its versatility and applicability to a wide range of information needs.

2.1 Basic Setup

In the TREC QA evaluations, an answer to a complex question consists of an unordered set of [document-id, answer string] pairs, where the strings are presumed to provide some relevant information that addresses the question. Although no explicit limit is placed on the length of the answer, the final metric penalizes verbosity (see below).

Evaluation of system output proceeds in two steps. First, answer strings from all submissions are gathered together and presented to a single assessor. The source of each answer string is blinded so that the assessor can not obviously tell which systems generated what output. Using these answers and searches performed during question development, the assessor creates a list of relevant nuggets. A nugget is a piece of information (i.e., "fact") that addresses one aspect of the user's question. Nuggets should be atomic, in the sense that an assessor should be able to make a binary decision as to whether the nugget appears in an answer string. Although a nugget represents a conceptual entity, the assessor provides a natural language description—primarily as a memory aid for the subsequent evaluation steps. These descriptions range from sentence-length document extracts to key phrases to telegraphic short-hand notes—their readability greatly varies from assessor to assessor.

r = # of *vital* nuggets returned
a = # of *okay* nuggets returned
R = # of *vital* nuggets in the answer key
l = # of non-whitespace characters in entire run

recall: $\mathcal{R} = r/R$

allowance: $\alpha = 100 \times (r + a)$

precision: $\mathcal{P} = \begin{cases} 1 & \text{if } l < \alpha \\ 1 - \frac{l-\alpha}{l} & \text{otherwise} \end{cases}$

$$F(\beta) = \frac{(\beta^2 + 1) \times \mathcal{P} \times \mathcal{R}}{\beta^2 \times \mathcal{P} + \mathcal{R}}$$

Figure 1: Official definition of F-score for nugget evaluation in TREC.

The assessor also manually classifies each nugget as either *vital* or *okay* (non-vital). Vital nuggets represent concepts that must be present in a "good" answer. Okay nuggets may contain interesting information, but are not essential.

In the second step, the same assessor who created the nuggets reads each system's output in turn and marks the appearance of the nuggets. An answer string contains a nugget if there is a *conceptual* match; that is, the match is independent of the particular wording used in the system's output. A nugget match is marked at most once per run—i.e., a system is not rewarded for retrieving a nugget multiple times. If the system's output contains more than one match for a nugget, the best match is selected and the rest are left unmarked. A single [document-id, answer string] pair in a system response can match 0, 1, or multiple nuggets.

The final F-score for an answer is calculated in the manner described in Figure 1, and the final score of a run is the average across the F-scores of all questions. The metric is a weighted harmonic mean between nugget precision and nugget recall, where recall is heavily favored (controlled by the β parameter, usually set to three). Nugget recall is calculated solely on vital nuggets, while nugget precision is approximated by a length allowance based on the number of both vital and okay nuggets returned. In an

earlier pilot study, researchers discovered that it was not possible for assessors to consistently enumerate the total set of nuggets contained in an answer, which corresponds to the denominator in a precision calculation (Voorhees, 2003). Thus, a penalty for verbosity serves as a surrogate for precision.

2.2 The Pyramid Extension

The vital/okay distinction has been identified as a weakness in the TREC nugget-based evaluation methodology (Hildebrandt et al., 2004; Lin and Demner-Fushman, 2005; Lin and Demner-Fushman, 2006). There do not appear to be any reliable indicators for predicting nugget importance, which makes it challenging to develop algorithms sensitive to this consideration. Since only vital nuggets affect nugget recall, it is difficult for systems to achieve non-zero scores on topics with few vital nuggets in the answer key. Thus, scores are easily affected by assessor errors and other random variations in evaluation conditions.

One direct consequence is that in previous TREC evaluations, the median score for many questions turned out to be zero. A binary distinction on nugget importance is insufficient to discriminate between the quality of runs that return no vital nuggets but different numbers of okay nuggets. Also, a score distribution heavily skewed towards zero makes meta-analyses of evaluation stability difficult to perform (Voorhees, 2005).

The pyramid extension (Lin and Demner-Fushman, 2006) was proposed to address the issues mentioned above. The idea was relatively simple: by soliciting vital/okay judgments from multiple assessors (after the list of nuggets has been produced by a primary assessor), it is possible to define nugget importance with greater granularity. Each nugget is assigned a weight between zero and one that is proportional to the number of assessors who judged it to be vital. Nugget recall from Figure 1 can be redefined to incorporate these weights:

$$\mathcal{R} = \frac{\sum_{m \in A} w_m}{\sum_{n \in V} w_n}$$

Where A is the set of reference nuggets that are matched in a system's output and V is the set of all reference nuggets; w_m and w_n are the weights of nuggets m and n, respectively.[1] The calculation of nugget precision remains the same.

3 Nugget Pyramids in TREC 2006

Lin and Demner-Fushman (2006) present experimental evidence in support of nugget pyramids by applying the proposal to results from previous TREC QA evaluations. Their simulation studies appear to support the assertion that pyramids address many of the issues raised in Section 2.2. Based on the results, NIST proceeded with a trial deployment of nugget pyramids in the TREC 2006 QA track. Although scores based on the binary vital/okay distinction were retained as the "official" metric, pyramid scores were simultaneously computed. This provided an opportunity to compare the two methodologies on a large scale.

3.1 The Data

The basic unit of evaluation for the main QA task at TREC 2006 was the "question series". Each series focused on a "target", which could be a person, organization, thing, or event. Individual questions in a series inquired about different facets of the target, and were explicitly classified as factoid, list, or Other. One complete series is shown in Figure 2. The Other questions can be best paraphrased as "Tell me interesting things about *X* that I haven't already explicitly asked about." It was the system's task to retrieve interesting nuggets about the target (in the opinion of the assessor), but credit was not given for retrieving facts already explicitly asked for in the factoid and list questions. The Other questions were evaluated using the nugget-based methodology, and are the subject of this analysis.

The QA test set in TREC 2006 contained 75 series. Of the 75 targets, 19 were persons, 19 were organizations, 19 were events, and 18 were things. The series contained a total of 75 Other questions (one per target). Each series contained 6–9 questions (counting the Other question), with most series containing 8 questions. The task employed the AQUAINT collection of newswire text (LDC catalog number LDC2002T31), consisting of English data drawn from three sources: the New York Times,

[1] Note that this new scoring model captures the existing binary vital/okay distinction in a straightforward way: vital nuggets get a score of one, and okay nuggets zero.

147	Britain's Prince Edward marries		
	147.1	FACTOID	When did Prince Edward engage to marry?
	147.2	FACTOID	Who did the Prince marry?
	147.3	FACTOID	Where did they honeymoon?
	147.4	FACTOID	Where was Edward in line for the throne at the time of the wedding?
	147.5	FACTOID	What was the Prince's occupation?
	147.6	FACTOID	How many people viewed the wedding on television?
	147.7	LIST	What individuals were at the wedding?
	147.8	OTHER	

Figure 2: Sample question series from TREC 2006.

Nugget	0	1	2	3	4	5	6	7	8
The couple had a long courtship	1	0	0	0	0	0	1	1	0
Queen Elizabeth II was delighted with the match	0	1	0	1	0	0	0	0	1
Queen named couple Earl and Contessa of Wessex	0	1	0	0	1	1	1	0	0
All marriages of Edward's siblings ended in divorce	0	0	0	0	0	1	0	0	1
Edward arranged for William to appear more cheerful in photo	0	0	0	0	0	0	0	0	0
they were married in St. Georges Chapel, Windsor	1	1	1	0	1	0	1	1	0

Figure 3: Multiple assessors' judgments of nugget importance for Series 147 (vital=1, okay=0). Assessor 2 was the same as the primary assessor (assessor 0), but judgments were elicited at different times.

the Associated Press, and the Xinhua News Service. There are approximately one million articles in the collection, totaling roughly three gigabytes. In total, 59 runs from 27 participants were submitted to NIST. For more details, see (Dang et al., 2006).

For the Other questions, nine sets of judgments were elicited from eight judges (the primary assessor who originally created the nuggets later annotated the nuggets once again). Each assessor was asked to assign the vital/okay label in a rapid fashion, without giving each decision much thought. Figure 3 gives an example of the multiple judgments for nuggets in Series 147. There is variation in notions of importance not only between different assessors, but also for a single assessor over time.

3.2 Results

After the human annotation process, nugget pyramids were built in the manner described by Lin and Demner-Fushman (2006). Two scores were computed for each run submitted to the TREC 2006 main QA task: one based on the vital/okay judgments of the primary assessor (which we call the binary F-score) and one based on the nugget pyramids (the pyramid F-score). The characteristics of the pyramid method can be inferred by comparing these two sets of scores.

Figure 4 plots the average binary and average pyramid F-scores for each run (which represents average performance across all series). Even though the nugget pyramid does not represent any single real user (a point we return to later), pyramid F-scores do correlate highly with the binary F-scores. The Pearson's correlation is 0.987, with a 95% confidence interval of [0.980, 1.00].

While the average F-score for a run is stable given a sufficient number of questions, the F-score for a single Other question exhibits greater variability across assessors. This is shown in Figure 5, which plots binary and pyramid F-scores for individual questions from all runs. In this case, the Pearson correlation is 0.870, with a 95% confidence interval of [0.863, 1.00].

For 16.4% of all Other questions, the nugget pyramid assigned a non-zero F-score where the original binary F-score was zero. This can be seen in the band of points on the left edge of the plot in Figure 5. This highlights the strength of nugget

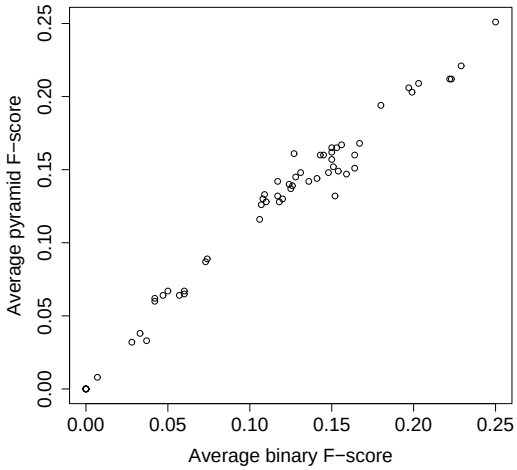


Figure 4: Scatter plot comparing the binary and pyramid F-scores for each run.

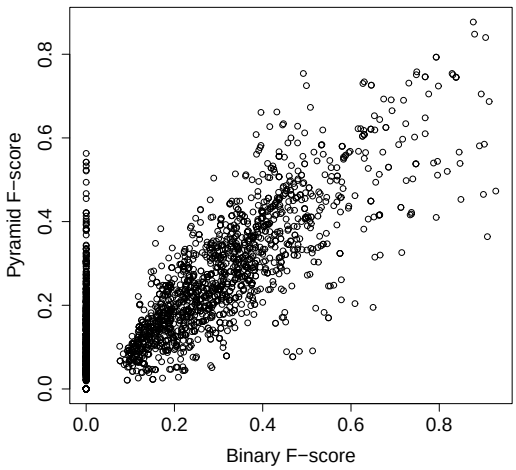


Figure 5: Scatter plot comparing the binary and pyramid F-scores for each Other question.

pyramids—their ability to smooth out assessor differences and more finely discriminate among system outputs. This is a key capability that is useful for system developers, particularly since algorithmic improvements are often incremental and small.

Because it is more stable than the single-assessor method of evaluation, the pyramid method also appears to have greater discriminative power. We fit a two-way analysis of variance model with the series and run as factors, and the binary F-score as the dependent variable. We found significant differences between series and between runs (p essentially equal to 0 for both factors). To determine which runs were significantly different from each other, we performed a multiple comparison using Tukey's honestly significant difference criterion and controlling for the experiment-wise Type I error so that the probability of declaring a difference between two runs to be significant, when it is actually not, is at most 5%. With 59 runs, there are $C_2^{59} = 1711$ different pairs that can be compared. The single-assessor method was able to declare one run to be significantly better than the other in 557 of these pairs. Using the pyramid F-scores, it was possible to find significant differences in performance between runs in 617 pairs.

3.3 Discussion

Any evaluation represents a compromise between effort (which correlates with cost) and insightfulness of results. The level of detail and meaningfulness of evaluations are constantly in tension with the availability of resources. Modifications to existing processes usually come at a cost that needs to be weighed against potential gains. Based on these considerations, the balance sheet for nugget pyramids shows a favorable orientation. In the TREC 2006 QA evaluation, soliciting vital/okay judgments from multiple assessors was not very time-consuming (a couple of hours per assessor). Analysis confirms that pyramid scores confer many benefits at an acceptable cost, thus arguing for its adoption in future evaluations.

Cost considerations precluded exploring other refinements to the nugget-based evaluation methodology. One possible alternative would involve asking multiple assessors to create different sets of nuggets from scratch. Not only would this be time-consuming, one would then need to deal with the additional complexities of aligning each assessor's nuggets list. This includes resolving issues such as nugget granularity, overlap in information content, implicature and other relations between nuggets, etc.

4 Exploration of Alternative Structures

Despite the demonstrated effectiveness of nugget pyramids, there are a few potential drawbacks that are worth discussing. One downside is that the nugget pyramid does not represent a single assessor. The nugget weights reflect the aggregation of opinions across a sample population, but there is no guar-

antee that the method for computing those weights actually captures any aspect of real user behavior. It can be argued that the binary F-score is more realistic since it reflects the opinion of a real user (the primary assessor), whereas the pyramid F-score tries to model the opinion of a mythical average user.

Although this point may seem somewhat counterintuitive, it represents a well-established tradition in the information retrieval literature (Voorhees, 2002). In document retrieval, for example, relevance judgments are provided by a single assessor—even though it is well known that there are large individual differences in notions of relevance. IR researchers believe that human idiosyncrasies are an inescapable fact present in any system designed for human users, and hence any attempt to remove those elements in the evaluation setup is actually undesirable. It is the responsibility of researchers to develop systems that are robust and flexible. This premise, however, does not mean that IR evaluation results are unstable or unreliable. Analyses have shown that despite large variations in human opinions, system rankings are remarkably stable (Voorhees, 2000; Sormunen, 2002)—that is, one can usually be confident about system comparisons.

The philosophy in IR sharply contrasts with work in NLP annotation tasks such as parsing, word sense disambiguation, and semantic role labeling—where researchers strive for high levels of interannotator agreement, often through elaborate guidelines. The difference in philosophies arises because unlike these NLP annotation tasks, where the products are used primarily by other NLP system components, IR (and likewise QA) is an end-user task. These systems are intended for real world use. Since people differ, systems must be able to accommodate these differences. Hence, there is a strong preference in QA for evaluations that maintain a model of the individual user.

4.1 Micro- vs. Macro-Averaging

The current nugget pyramid method leverages multiple judgments to define a weight for each individual nugget, and then incorporates this weight into the F-score computation. As an alternative, we propose another method for combining the opinions of multiple assessors: evaluate system responses individually against N sets of binary judgments, and then compute the mean across those scores. We define the macro-averaged binary F-score over a set $A = \{a_1, ..., a_N\}$ of N assessors as:

$$\mathcal{F} = \frac{\sum_{a \in A} F_a}{N}$$

Where F_a is the binary F-score according to the vital/okay judgments of assessor a. The differences between the pyramid F-score and the macro-averaged binary F-score correspond to the distinction between micro- and macro-averaging discussed in the context of text classification (Lewis, 1991). In those applications, both measures are meaningful depending on focus: individual instances or entire classes. In tasks where it is important to correctly classify individual instances, micro-averaging is more appropriate. In tasks where it is important to correctly identify a class, macro-averaging better quantifies performance. In classification tasks, imbalance in the prevalence of each class can lead to large differences in macro- and micro-averaged scores. Analogizing to our work, the original formulation of nugget pyramids corresponds to micro-averaging (since we focus on individual nuggets), while the alternative corresponds to macro-averaging (since we focus on the assessor).

We additionally note that the two methods encode different assumptions. Macro-averaging assumes that there is nothing intrinsically interesting about a nugget—it is simply a matter of a particular user with particular needs finding a particular nugget to be of interest. Micro-averaging, on the other hand, assumes that some nuggets are inherently interesting, independent of the particular interests of users.[2]

Each approach has characteristics that make it desirable. From the perspective of evaluators, the macro-averaged binary F-score is preferable because it models real users; each set of binary judgments represents the information need of a real user, each binary F-score represents how well an answer will satisfy a real user, and the macro-averaged binary F-score represents how well an answer will satisfy, on average, a sample population of real users. From the perspective of QA system developers, the micro-averaged nugget pyramid F-score is preferable because it allows finer discrimination in in-

[2] We are grateful to an anonymous reviewer for this insight.

dividual nugget performance, which enables better techniques for system training and optimization.

The macro-averaged binary F-score has the same desirable properties as the micro-averaged pyramid F-score in that fewer responses will have zero F-scores as compared to the single-assessor binary F-score. We demonstrate this as follows. Let X be a response that receives a non-zero pyramid F-score. Let $A = \{a_1, a_2, a_3, ..., a_N\}$ be the set of N assessors. Then it can be proven that X also receives a non-zero macro-averaged binary F-score:

1. There exists some nugget v with weight greater than 0, such that an answer string r in X matches v. *(def. of pyramid recall)*
2. There exists some assessor $a_p \in A$ who marked v as vital. *(def. of pyramid nugget weight)*
3. To show that X will also receive a non-zero macro-averaged binary score, it is sufficient to show that there is some assessor $a_m \in A$ such that X receives a non-zero F-score when evaluated using just the vital/okay judgments of a_m. *(def. of macro-averaged binary F-score)*
4. But, such an assessor does exist, namely assessor a_p: Consider the binary F-score assigned to X according to just assessor a_p. The recall of X is greater than zero, since X contains the response r that matches the nugget v that was marked as vital by a_p *(from (2), (1), and the def. of recall)*. The precision must also be greater than zero *(def. of precision)*. Therefore, the macro-averaged binary F-score of X is non-zero. *(def. of F-score)*

4.2 Analysis from TREC 2006

While the macro-averaged method is guaranteed to produce no more zero-valued scores than the micro-averaged pyramid method, it is not guaranteed that the scores will be the same for any given response. What are the empirical characteristics of each approach? To explore this question, we once again examined data from TREC 2006.

Figure 6 shows a scatter plot of the pyramid F-score and macro-averaged binary F-score for every Other questions in all runs from the TREC 2006 QA track main task. Despite focusing on different aspects of the evaluation setup, these measures

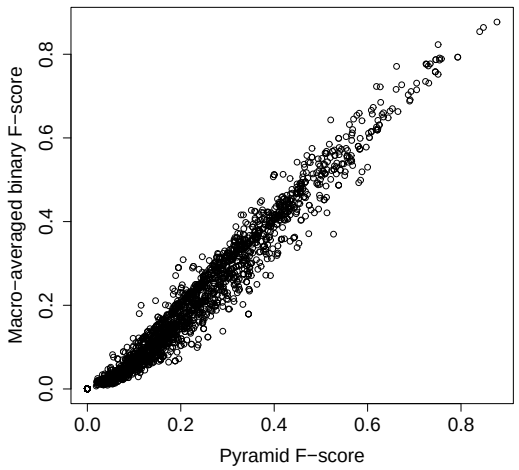


Figure 6: Scatter plot comparing the pyramid and macro-averaged binary F-scores for all questions.

	binary	micro	macro
binary	1.000/1.000	0.870/0.987	0.861/0.988
micro	-	1.000/1.000	0.985/0.996
macro	-	-	1.000/1.000

Table 1: Pearson's correlation of F-scores, by question and by run.

are highly correlated, even at the level of individual questions. Table 1 provides a summary of the correlations between the original binary F-score, the (micro-averaged) pyramid F-score, and the macro-averaged binary F-score. Pearson's r is given for F-scores at the individual question level (first number) and at the run level (second number). The correlation between all three variants are about equal at the level of system runs. At the level of individual questions, the micro- and macro-averaged F-scores (using multiple judgments) are still highly correlated with each other, but each is less correlated with the single-assessor binary F-score.

4.3 Discussion

The differences between macro- and micro-averaging methods invoke a more general discussion on notions of nugget importance. There are actually two different issues we are attempting to address with our different approaches: the first is a more granular scale of nugget importance, the second is variations across a population of users. In

the micro-averaged pyramid F-scores, we achieve the first by leveraging the second, i.e., binary judgments from a large population are combined to yield weights for individual nuggets. In the macro-averaged binary F-score, we focus solely on population effects without addressing granularity of nugget importance.

Exploring this thread of argument, we can formulate additional approaches for tackling these issues. We could, for example, solicit more granular individual judgments on each nugget from each assessor, perhaps on a Likert scale or as a continuous quantity ranging from zero to one. This would yield two more methods for computing F-scores, both a macro-averaged and a micro-averaged variant. The macro-averaged variant would be especially attractive because it reflects real users and yet individual F-scores remain discriminative. Despite its possible advantages, this extension is rejected based on resource considerations; making snap binary judgments on individual nuggets is much quicker than a multi-scaled value assignment—at least at present, the additional costs are not sufficient to offset the potential gains.

5 Conclusion

The important role that large-scale evaluations play in guiding research in human language technologies means that the community must "get it right." This would ordinarily call for a more conservative approach to avoid changes that might have unintended consequences. However, evaluation methodologies must evolve to reflect the shifting interests of the research community to remain relevant. Thus, organizers of evaluations must walk a fine line between progress and chaos. Nevertheless, the introduction of nugget pyramids in the TREC QA evaluation provides a case study showing how this fine balance can indeed be achieved. The addition of multiple judgments of nugget importance yields an evaluation that is both more stable and more discriminative than the original single-assessor evaluation, while requiring only a small additional cost in terms of human labor.

We have explored two different methods for combining judgments from multiple assessors to address shortcomings in the original nugget-based evaluation setup. Although they make different assumptions about the evaluation, results from both approaches are highly correlated. Thus, we can continue employing the pyramid-based method, which is well-suited for developing systems, and still be assured that the results remain consistent with an evaluation method that maintains a model of real individual users.

Acknowledgments

This work has been supported in part by DARPA contract HR0011-06-2-0001 (GALE). The second author would like to thank Kiri and Esther for their kind support.

References

H. Dang, J. Lin, and D. Kelly. 2006. Overview of the TREC 2006 question answering track. In *Proc. of TREC 2006*.

W. Hildebrandt, B. Katz, and J. Lin. 2004. Answering definition questions with multiple knowledge sources. In *Proc. HLT/NAACL 2004*.

D. Lewis. 1991. Evaluating text categorization. In *Proc. of the Speech and Natural Language Workshop*.

J. Lin and D. Demner-Fushman. 2005. Automatically evaluating answers to definition questions. In *Proc. of HLT/EMNLP 2005*.

J. Lin and D. Demner-Fushman. 2006. Will pyramids built of nuggets topple over? In *Proc. of HLT/NAACL 2006*.

E. Sormunen. 2002. Liberal relevance criteria of TREC—counting on negligible documents? In *Proc. of SIGIR 2002*.

E. Voorhees. 2000. Variations in relevance judgments and the measurement of retrieval effectiveness. *IP&M*, 36(5):697–716.

E. Voorhees. 2002. The philosophy of information retrieval evaluation. In *Proc. of CLEF Workshop*.

E. Voorhees. 2003. Overview of the TREC 2003 question answering track. In *Proc. of TREC 2003*.

E. Voorhees. 2005. Using question series to evaluate question answering system effectiveness. In *Proc. of HLT/EMNLP 2005*.

Exploiting Syntactic and Shallow Semantic Kernels for Question/Answer Classification

Alessandro Moschitti
University of Trento
38050 Povo di Trento
Italy
moschitti@dit.unitn.it

Silvia Quarteroni
The University of York
York YO10 5DD
United Kingdom
silvia@cs.york.ac.uk

Roberto Basili
"Tor Vergata" University
Via del Politecnico 1
00133 Rome, Italy
basili@info.uniroma2.it

Suresh Manandhar
The University of York
York YO10 5DD
United Kingdom
suresh@cs.york.ac.uk

Abstract

We study the impact of syntactic and shallow semantic information in automatic classification of questions and answers and answer re-ranking. We define (a) new tree structures based on shallow semantics encoded in Predicate Argument Structures (PASs) and (b) new kernel functions to exploit the representational power of such structures with Support Vector Machines. Our experiments suggest that syntactic information helps tasks such as question/answer classification and that shallow semantics gives remarkable contribution when a reliable set of PASs can be extracted, e.g. from answers.

1 Introduction

Question answering (QA) is as a form of information retrieval where one or more answers are returned to a question in natural language in the form of sentences or phrases. The typical QA system architecture consists of three phases: question processing, document retrieval and answer extraction (Kwok et al., 2001).

Question processing is often centered on question classification, which selects one of k expected answer classes. Most accurate models apply supervised machine learning techniques, e.g. SNoW (Li and Roth, 2005), where questions are encoded using various lexical, syntactic and semantic features. The retrieval and answer extraction phases consist in retrieving relevant documents (Collins-Thompson et al., 2004) and selecting candidate answer passages from them. A further answer re-ranking phase is optionally applied. Here, too, the syntactic structure of a sentence appears to provide more useful information than a bag of words (Chen et al., 2006), although the correct way to exploit it is still an open problem.

An effective way to integrate syntactic structures in machine learning algorithms is the use of tree kernel (TK) functions (Collins and Duffy, 2002), which have been successfully applied to question classification (Zhang and Lee, 2003; Moschitti, 2006) and other tasks, e.g. relation extraction (Zelenko et al., 2003; Moschitti, 2006). In more complex tasks such as computing the relatedness between questions and answers in answer re-ranking, to our knowledge no study uses kernel functions to encode syntactic information. Moreover, the study of shallow semantic information such as predicate argument structures annotated in the PropBank (PB) project (Kingsbury and Palmer, 2002) (www.cis.upenn.edu/~ace) is a promising research direction. We argue that semantic structures can be used to characterize the relation between a question and a candidate answer.

In this paper, we extensively study new structural representations, encoding parse trees, bag-of-words, POS tags and predicate argument structures (PASs) for question classification and answer re-ranking. We define new tree representations for both simple and nested PASs, i.e. PASs whose arguments are other predicates (Section 2). Moreover, we define new kernel functions to exploit PASs, which we automatically derive with our SRL system (Moschitti et al., 2005) (Section 3).

Our experiments using SVMs and the above ker-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 776–783,
Prague, Czech Republic, June 2007.

nels and data (Section 4) shows the following: (a) our approach reaches state-of-the-art accuracy on question classification. (b) PB predicative structures are not effective for question classification but show promising results for answer classification on a corpus of answers to TREC-QA 2001 description questions. We created such dataset by using YourQA (Quarteroni and Manandhar, 2006), our basic Web-based QA system[1]. (c) The answer classifier increases the ranking accuracy of our QA system by about 25%.

Our results show that PAS and syntactic parsing are promising methods to address tasks affected by data sparseness like question/answer categorization.

2 Encoding Shallow Semantic Structures

Traditionally, information retrieval techniques are based on the *bag-of-words* (BOW) approach augmented by language modeling (Allan et al., 2002). When the task requires the use of more complex semantics, the above approaches are often inadequate to perform fine-level textual analysis.

An improvement on BOW is given by the use of syntactic parse trees, e.g. for question classification (Zhang and Lee, 2003), but these, too are inadequate when dealing with definitional answers expressed by long and articulated sentences or even paragraphs. On the contrary, shallow semantic representations, bearing a more "compact" information, could prevent the sparseness of deep structural approaches and the weakness of BOW models.

Initiatives such as PropBank (PB) (Kingsbury and Palmer, 2002) have made possible the design of accurate automatic Semantic Role Labeling (SRL) systems (Carreras and Màrquez, 2005). Attempting an application of SRL to QA hence seems natural, as pinpointing the answer to a question relies on a deep understanding of the semantics of both.
Let us consider the PB annotation: [$_{ARG1}$ Antigens] were [$_{AM-TMP}$ originally] [$_{rel}$ defined] [$_{ARG2}$ as non-self molecules].
Such annotation can be used to design a shallow semantic representation that can be matched against other semantically similar sentences, e.g. [$_{ARG0}$ Researchers] [$_{rel}$ describe] [$_{ARG1}$ antigens] [$_{ARG2}$ as foreign molecules] [$_{ARGM-LOC}$ in the body].
For this purpose, we can represent the above annotated sentences using the tree structures described in Figure 1. In this compact representation, hereafter Predicate-Argument Structures (PAS), arguments are replaced with their most important word – often referred to as the semantic head. This reduces data sparseness with respect to a typical BOW representation.

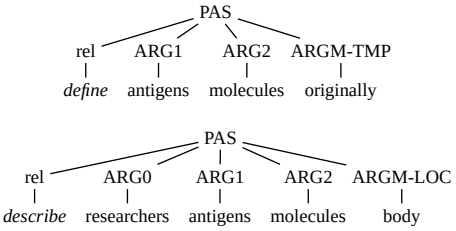


Figure 1: Compact predicate argument structures of two different sentences.

However, sentences rarely contain a single predicate; it happens more generally that propositions contain one or more subordinate clauses. For instance let us consider a slight modification of the first sentence: "Antigens were originally defined as non-self molecules *which bound specifically to antibodies*[2]." Here, the main predicate is "defined", followed by a subordinate predicate "bound". Our SRL system outputs the following *two* annotations:
(1) [$_{ARG1}$ Antigens] were [$_{ARGM-TMP}$ originally] [$_{rel}$ defined] [$_{ARG2}$ as non-self molecules which bound specifically to antibodies].
(2) Antigens were originally defined as [$_{ARG1}$ non-self molecules] [$_{R-A1}$ which] [$_{rel}$ bound] [$_{ARGM-MNR}$ specifically] [$_{ARG2}$ to antibodies].
giving the PASs in Figure 2.(a) resp. 2.(b).

As visible in Figure 2.(a), when an argument node corresponds to an entire subordinate clause, we label its leaf with PAS, e.g. the leaf of ARG2. Such PAS node is actually the root of the subordinate clause in Figure 2.(b). Taken as standalone, such PASs do not express the whole meaning of the sentence; it is more accurate to define a single structure encoding the dependency between the two predicates as in

[1]Demo at: http://cs.york.ac.uk/aig/aqua.

[2]This is an actual answer to "What are antibodies?" from our question answering system, YourQA.

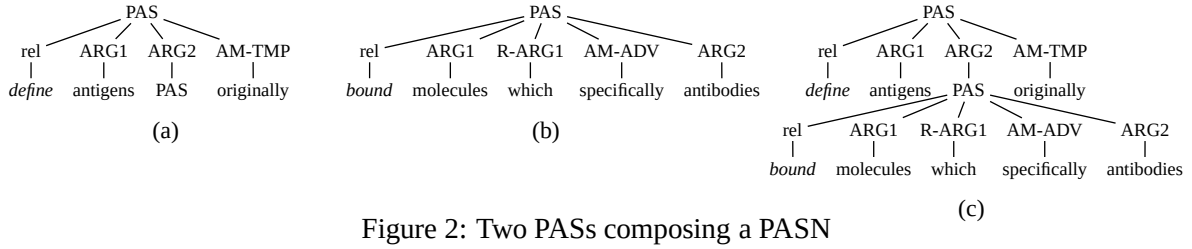


Figure 2: Two PASs composing a PASN

Figure 2.(c). We refer to nested PASs as PASNs.

It is worth to note that semantically equivalent sentences syntactically expressed in different ways share the same PB arguments and the same PASs, whereas semantically different sentences result in different PASs. For example, the sentence: "Antigens were originally defined as *antibodies* which bound specifically to *non-self molecules*", uses the same words as (2) but has different meaning. Its PB annotation:

(3) `Antigens were originally defined as [`$_{ARG1}$ `antibodies] [`$_{R-A1}$ `which] [`$_{rel}$ `bound] [`$_{ARGM-MNR}$ `specifically] [`$_{ARG2}$ `to non-self molecules]`,

clearly differs from (2), as ARG2 is now *non-self molecules*; consequently, the PASs are also different.

Once we have assumed that parse trees and PASs can improve on the simple BOW representation, we face the problem of representing tree structures in learning machines. Section 3 introduces a viable approach based on tree kernels.

3 Syntactic and Semantic Kernels for Text

As mentioned above, encoding syntactic/semantic information represented by means of tree structures in the learning algorithm is problematic. A first solution is to use all its possible substructures as features. Given the combinatorial explosion of considering subparts, the resulting feature space is usually very large. A tree kernel (TK) function which computes the number of common subtrees between two syntactic parse trees has been given in (Collins and Duffy, 2002). Unfortunately, such subtrees are subject to the constraint that their nodes are taken with all or none of the children they have in the original tree. This makes the TK function not well suited for the PAS trees defined above. For instance, although the two PASs of Figure 1 share most of the subtrees rooted in the PAS node, Collins and Duffy's kernel would compute no match.

In the next section we describe a new kernel derived from the above tree kernel, able to evaluate the meaningful substructures for PAS trees. Moreover, as a single PAS may not be sufficient for text representation, we propose a new kernel that combines the contributions of different PASs.

3.1 Tree kernels

Given two trees T_1 and T_2, let $\{f_1, f_2, ..\} = \mathcal{F}$ be the set of substructures (fragments) and $I_i(n)$ be equal to 1 if f_i is rooted at node n, 0 otherwise. Collins and Duffy's kernel is defined as

$$TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2), \quad (1)$$

where N_{T_1} and N_{T_2} are the sets of nodes in T_1 and T_2, respectively and $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} I_i(n_1) I_i(n_2)$. The latter is equal to the number of common fragments rooted in nodes n_1 and n_2. Δ can be computed as follows:

(1) if the productions (i.e. the nodes with their direct children) at n_1 and n_2 are different then $\Delta(n_1, n_2) = 0$;

(2) if the productions at n_1 and n_2 are the same, and n_1 and n_2 only have leaf children (i.e. they are pre-terminal symbols) then $\Delta(n_1, n_2) = 1$;

(3) if the productions at n_1 and n_2 are the same, and n_1 and n_2 are not pre-terminals then $\Delta(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (1 + \Delta(c_{n_1}^j, c_{n_2}^j))$, where $nc(n_1)$ is the number of children of n_1 and c_n^j is the j-th child of n.

Such tree kernel can be normalized and a λ factor can be added to reduce the weight of large structures (refer to (Collins and Duffy, 2002) for a complete description). The critical aspect of steps (1), (2) and (3) is that the productions of two evaluated nodes have to be identical to allow the match of further descendants. This means that common substructures cannot be composed by a node with only some of its

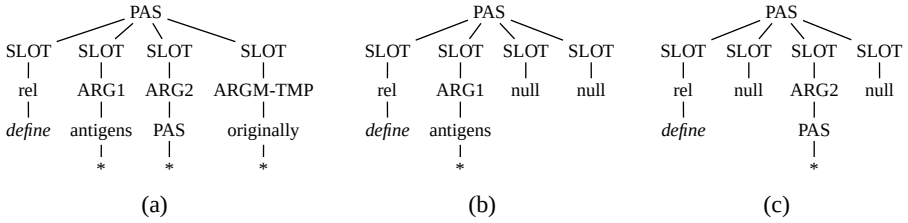


Figure 3: A PAS with some of its fragments.

children as an effective PAS representation would require. We solve this problem by designing the Shallow Semantic Tree Kernel (SSTK) which allows to match portions of a PAS.

3.2 The Shallow Semantic Tree Kernel (SSTK)

The SSTK is based on two ideas: first, we change the PAS, as shown in Figure 3.(a) by adding *SLOT* nodes. These accommodate argument labels in a specific order, i.e. we provide a fixed number of slots, possibly filled with *null* arguments, that encode all possible predicate arguments. For simplicity, the figure shows a structure of just 4 arguments, but more can be added to accommodate the maximum number of arguments a predicate can have. Leaf nodes are filled with the wildcard character * but they may alternatively accommodate additional information.

The slot nodes are used in such a way that the adopted TK function can generate fragments containing one or more children like for example those shown in frames (b) and (c) of Figure 3. As previously pointed out, if the arguments were directly attached to the root node, the kernel function would only generate the structure with all children (or the structure with no children, i.e. empty).

Second, as the original tree kernel would generate many matches with slots filled with the null label, we have set a new step 0:

(0) if n_1 (or n_2) is a pre-terminal node and its child label is *null*, $\Delta(n_1, n_2) = 0$;

and subtract one unit to $\Delta(n_1, n_2)$, in step 3:

(3) $\Delta(n_1, n_2) = \prod_{j=1}^{nc(n_1)}(1 + \Delta(c_{n_1}^j, c_{n_2}^j)) - 1$,

The above changes generate a new Δ which, when substituted (in place of the original Δ) in Eq. 1, gives the new Shallow Semantic Tree Kernel. To show that SSTK is effective in counting the number of relations shared by two PASs, we propose the following:

Proposition 1 *The new Δ function applied to the modified PAS counts the number of all possible k-ary relations derivable from a set of k arguments, i.e. $\sum_{i=1}^{k}\binom{k}{i}$ relations of arity from 1 to k (the predicate being considered as a special argument).*

Proof We observe that a kernel applied to a tree and itself computes all its substructures, thus if we evaluate SSTK between a PAS and itself we must obtain the number of generated k-ary relations. We prove by induction the above claim.

For the base case ($k = 0$): we use a PAS with no arguments, i.e. all its slots are filled with null labels. Let r be the PAS root; since r is not a pre-terminal, step 3 is selected and Δ is recursively applied to all r's children, i.e. the slot nodes. For the latter, step 0 assigns $\Delta(c_r^j, c_r^j) = 0$. As a result, $\Delta(r, r) = \prod_{j=1}^{nc(r)}(1 + 0) - 1 = 0$ and the base case holds.

For the general case, r is the root of a PAS with $k+1$ arguments. $\Delta(r, r) = \prod_{j=1}^{nc(r)}(1 + \Delta(c_r^j, c_r^j)) - 1$ $=\prod_{j=1}^{k}(1+\Delta(c_r^j, c_r^j))\times(1+\Delta(c_r^{k+1}, c_r^{k+1}))-1$. For k arguments, we assume by induction that $\prod_{j=1}^{k}(1+\Delta(c_r^j, c_r^j)) - 1 = \sum_{i=1}^{k}\binom{k}{i}$, i.e. the number of k-ary relations. Moreover, $(1 + \Delta(c_r^{k+1}, c_r^{k+1})) = 2$, thus $\Delta(r, r) = \sum_{i=1}^{k}\binom{k}{i} \times 2 = 2^k \times 2 = 2^{k+1} = \sum_{i=1}^{k+1}\binom{k+1}{i}$, i.e. all the relations until arity $k + 1$ □

TK functions can be applied to sentence parse trees, therefore their usefulness for text processing applications, e.g. question classification, is evident. On the contrary, the SSTK applied to one PAS extracted from a text fragment may not be meaningful since its representation needs to take into account all the PASs that it contains. We address such problem

by defining a kernel on *multiple* PASs.

Let P_t and $P_{t'}$ be the sets of PASs extracted from the text fragment t and t'. We define:

$$K_{\texttt{all}}(P_t, P_{t'}) = \sum_{p \in P_t} \sum_{p' \in P_{t'}} SSTK(p, p'), \quad (2)$$

While during the experiments (Sect. 4) the $K_{\texttt{all}}$ kernel is used to handle predicate argument structures, TK (Eq. 1) is used to process parse trees and the linear kernel to handle POS and BOW features.

4 Experiments

The purpose of our experiments is to study the impact of the new representations introduced earlier for QA tasks. In particular, we focus on question classification and answer re-ranking for Web-based QA systems.

In the question classification task, we extend previous studies, e.g. (Zhang and Lee, 2003; Moschitti, 2006), by testing a set of previously designed kernels and their combination with our new Shallow Semantic Tree Kernel. In the answer re-ranking task, we approach the problem of detecting description answers, among the most complex in the literature (Cui et al., 2005; Kazawa et al., 2001).

The representations that we adopt are: bag-of-words (BOW), bag-of-POS tags (POS), parse tree (PT), predicate argument structure (PAS) and nested PAS (PASN). BOW and POS are processed by means of a linear kernel, PT is processed with TK, PAS and PASN are processed by SSTK. We implemented the proposed kernels in the SVM-light-TK software available at `ai-nlp.info.uniroma2.it/moschitti/` which encodes tree kernel functions in SVM-light (Joachims, 1999).

4.1 Question classification

As a first experiment, we focus on question classification, for which benchmarks and baseline results are available (Zhang and Lee, 2003; Li and Roth, 2005). We design a question multi-classifier by combining n binary SVMs[3] according to the ONE-vs-ALL scheme, where the final output class is the one associated with the most probable prediction. The PASs were automatically derived by our SRL system which achieves a 76% F1-measure (Moschitti et al., 2005).

As benchmark data, we use the question training and test set available at: `l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/`, where the test set are the 500 TREC 2001 test questions (Voorhees, 2001). We refer to this split as UIUC. The performance of the multi-classifier and the individual binary classifiers is measured with accuracy resp. F1-measure. To collect statistically significant information, we run 10-fold cross validation on the 6,000 questions.

Features	Accuracy (UIUC)	Accuracy (c.v.)
PT	90.4	84.8±1.2
BOW	90.6	84.7±1.2
PAS	34.2	43.0±1.9
POS	26.4	32.4±2.1
PT+BOW	**91.8**	**86.1±1.1**
PT+BOW+POS	91.8	84.7±1.5
PAS+BOW	90.0	82.1±1.3
PAS+BOW+POS	88.8	81.0±1.5

Table 1: Accuracy of the question classifier with different feature combinations

Question classification results Table 1 shows the accuracy of different question representations on the UIUC split (Column 1) and the average accuracy ± the corresponding confidence limit (at 90% significance) on the cross validation splits (Column 2).(i) The TK on PT and the linear kernel on BOW produce a very high result, i.e. about 90.5%. This is higher than the best outcome derived in (Zhang and Lee, 2003), i.e. 90%, obtained with a kernel combining BOW and PT on the same data. Combined with PT, BOW reaches 91.8%, very close to the 92.5% accuracy reached in (Li and Roth, 2005) using complex semantic information from external resources.
(ii) The PAS feature provides no improvement. This is mainly because at least half of the training and test questions only contain the predicate "to be", for which a PAS cannot be derived by a PB-based shallow semantic parser.
(iii) The 10-fold cross-validation experiments confirm the trends observed in the UIUC split. The best model (according to statistical significance) is PT+BOW, achieving an 86.1% average accuracy[4].

[3] We adopted the default regularization parameter (i.e., the average of $1/||\vec{x}||$) and tried a few cost-factor values to adjust the rate between Precision and Recall on the development set.

[4] This value is lower than the UIUC split one as the UIUC test set is not consistent with the training set (it contains the

4.2 Answer classification

Question classification does not allow to fully exploit the PAS potential since questions tend to be short and with few verbal predicates (i.e. the only ones that our SRL system can extract). A different scenario is answer classification, i.e. deciding if a passage/sentence correctly answers a question. Here, the semantics to be generated by the classifier are not constrained to a small taxonomy and answer length may make the PT-based representation too sparse.

We learn answer classification with a binary SVM which determines if an answer is correct for the target question: here, the classification instances are ⟨question, answer⟩ pairs. Each pair component can be encoded with PT, BOW, PAS and PASN representations (processed by previous kernels).

As test data, we collected the 138 TREC 2001 test questions labeled as "description" and for each, we obtained a list of answer paragraphs extracted from Web documents using YourQA. Each paragraph sentence was manually evaluated based on whether it contained an answer to the corresponding question. Moreover, to simplify the classification problem, we isolated for each paragraph the sentence which obtained the maximal judgment (in case more than one sentence in the paragraph had the same judgment, we chose the first one). We collected a corpus containing 1309 sentences, 416 of which – labeled "+1" – answered the question either concisely or with noise; the rest – labeled "-1"– were either irrelevant to the question or contained hints relating to the question but could not be judged as valid answers[5].

Answer classification results To test the impact of our models on answer classification, we ran 5-fold cross-validation, with the constraint that two pairs $\langle q, a_1 \rangle$ and $\langle q, a_2 \rangle$ associated with the same question q could not be split between training and testing. Hence, each reported value is the average over 5 different outcomes. The standard deviations ranged

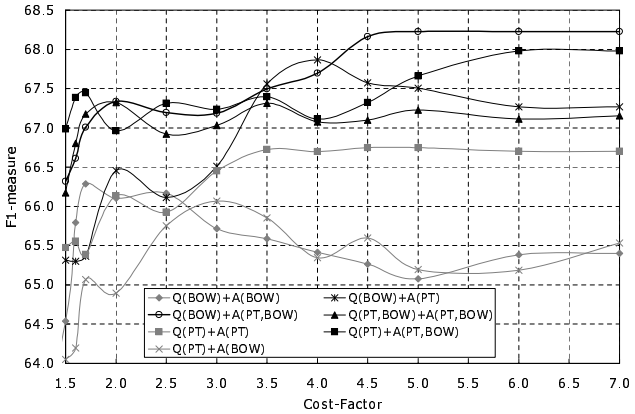


Figure 4: Impact of the BOW and PT features on answer classification

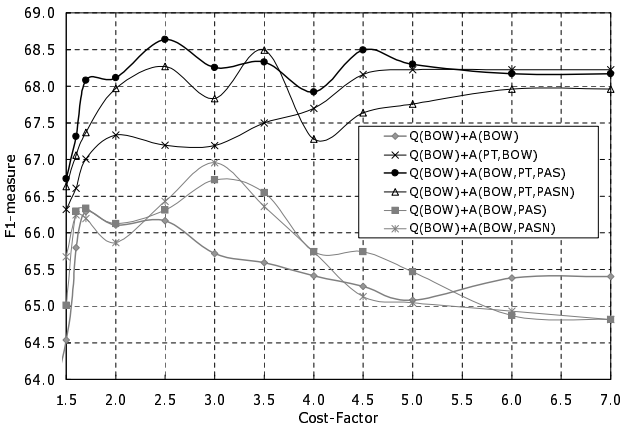


Figure 5: Impact of the PAS and PASN features combined with the BOW and PT features on answer classification

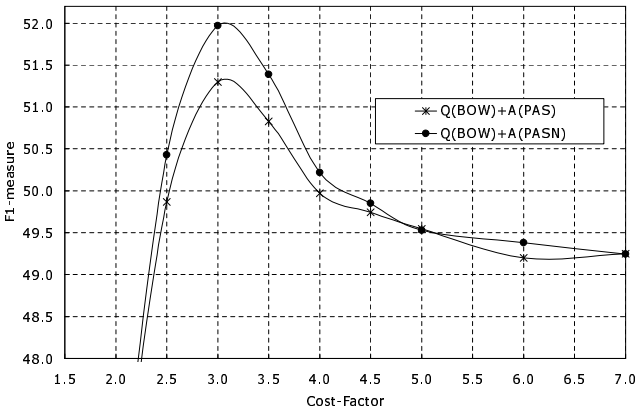


Figure 6: Comparison between PAS and PASN when used as standalone features for the answer on answer classification

TREC 2001 questions) and includes a larger percentage of easily classified question types, e.g. the numeric (22.6%) and description classes (27.6%) whose percentage in training is 16.4% resp. 16.2%.

[5]For instance, given the question "What are invertebrates?", the sentence "At least 99% of all animal species are invertebrates, comprising . . . " was labeled "-1" , while "Invertebrates are animals without backbones." was labeled "+1".

approximately between 2.5 and 5. The experiments were organized as follows:

First, we examined the contributions of BOW and PT representations as they proved very important for question classification. Figure 4 reports the plot of the F1-measure of answer classifiers trained with all combinations of the above models according to different values of the cost-factor parameter, adjusting the rate between Precision and Recall. We see here that the most accurate classifiers are the ones using both the answer's BOW and PT feature and either the question's PT or BOW feature (i.e. Q(BOW) + A(PT,BOW) resp. Q(PT) + A(PT,BOW) combinations). When PT is used for the answer the simple BOW model is outperformed by 2 to 3 points. Hence, we infer that both the answer's PT and BOW features are very useful in the classification task. However, PT does not seem to provide additional information to BOW when used for question representation. This can be explained by considering that answer classification (restricted to description questions) does not require question type classification since its main purpose is to detect question/answer relations. In this scenario, the question's syntactic structure does not seem to provide much more information than BOW.

Secondly, we evaluated the impact of the newly defined PAS and PASN features combined with the best performing previous model, i.e. Q(BOW) + A(PT,BOW). Figure 5 illustrates the F1-measure plots again according to the cost-factor parameter. We observe here that model Q(BOW) + A(PT,BOW,PAS) greatly outperforms model Q(BOW) + A(PT,BOW), proving that the PAS feature is very useful for answer classification, i.e. the improvement is about 2 to 3 points while the difference with the BOW model, i.e. Q(BOW) + A(BOW), exceeds 3 points. The Q(BOW) + A(PT,BOW,PASN) model is not more effective than Q(BOW) + A(PT,BOW,PAS). This suggests either that PAS is more effective than PASN or that when the PT information is added, the PASN contribution fades out.

To further investigate the previous issue, we finally compared the contribution of the PAS and PASN when combined with the question's BOW feature alone, i.e. no PT is used. The results, reported in Figure 6, show that this time PASN performs better than PAS. This suggests that the dependencies between the nested PASs are in some way captured by the PT information. Indeed, it should be noted that we join predicates only in case one is subordinate to the other, thus considering only a restricted set of all possible predicate dependencies. However, the improvement over PAS confirms that PASN is the right direction to encode shallow semantics from different sentence predicates.

Baseline	P	R	F1-measure
Gg@5	39.22±3.59	33.15±4.22	35.92±3.95
QA@5	39.72±3.44	34.22±3.63	36.76±3.56
Gg@all	31.58±0.58	100	48.02±0.67
QA@all	31.58±0.58	100	48.02±0.67
	Gg	QA	Re-ranker
MRR	48.97±3.77	56.21±3.18	81.12±2.12

Table 2: Baseline classifiers accuracy and MRR of YourQA (QA), Google (Gg) and the best re-ranker

4.3 Answer re-ranking

The output of the answer classifier can be used to re-rank the list of candidate answers of a QA system. Starting from the top answer, each instance can be classified based on its correctness with respect to the question. If it is classified as correct its rank is unchanged; otherwise it is pushed down, until a lower ranked incorrect answer is found.

We used the answer classifier with the highest F1-measure on the development set according to different cost-factor values[6]. We applied such model to the Google ranks and to the ranks of our Web-based QA system, i.e. YourQA. The latter uses Web documents corresponding to the top 20 Google results for the question. Then, each sentence in each document is compared to the question via a blend of similarity metrics used in the answer extraction phase to select the most relevant sentence. A passage of up to 750 bytes is then created around the sentence and returned as an answer.

Table 2 illustrates the results of the answer classifiers derived by exploiting Google (Gg) and YourQA (QA) ranks: the top N ranked results are considered as correct definitions and the remaining ones as in-

[6]However, by observing the curves in Fig. 5, the selected parameters appear as pessimistic estimates for the best model improvement: the one for BOW is the absolute maximum, but an average one is selected for the best model.

correct for different values of N. We show N = 5 and the maximum N (all), i.e. all the available answers. Each measure is the average of the Precision, Recall and F1-measure from cross validation. The F1-measure of Google and YourQA are greatly outperformed by our answer classifier.

The last row of Table 2 reports the MRR[7] achieved by Google, YourQA (QA) and YourQA after re-ranking (Re-ranker). We note that Google is outperformed by YourQA since its ranks are based on whole documents, not on single passages. Thus Google may rank a document containing several sparsely distributed question words higher than documents with several words concentrated in one passage, which are more interesting. When the answer classifier is applied to improve the YourQA ranking, the MRR reaches 81.1%, rising by about 25%.

Finally, it is worth to note that the answer classifier based on Q(BOW)+A(BOW,PT,PAS) model (parameterized as described) gave a 4% higher MRR than the one based on the simple BOW features. As an example, for question "What is foreclosure?", the sentence "Foreclosure means that the lender takes possession of your home and sells it in order to get its money back." was correctly classified by the best model, while BOW failed.

5 Conclusion

In this paper, we have introduced new structures to represent textual information in three question answering tasks: question classification, answer classification and answer re-ranking. We have defined tree structures (PAS and PASN) to represent predicate-argument relations, which we automatically extract using our SRL system. We have also introduced two functions, $SSTK$ and $K_{\texttt{all}}$, to exploit their representative power.

Our experiments with SVMs and the above models suggest that syntactic information helps tasks such as question classification whereas semantic information contained in PAS and PASN gives promising results in answer classification.

In the future, we aim to study ways to capture relations between predicates so that more general semantics can be encoded by PASN. Forms of generalization for predicates and arguments within PASNs like LSA clusters, WordNet synsets and FrameNet (roles and frames) information also appear as a promising research area.

[7]The Mean Reciprocal Rank is defined as: $MRR = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{rank_i}$, where n is the number of questions and $rank_i$ is the rank of the first correct answer to question i.

Acknowledgments

We thank the anonymous reviewers for their helpful suggestions. Alessandro Moschitti would like to thank the AMI2 lab at the University of Trento and the EU project LUNA "spoken Language UNderstanding in multilinguAl communication systems" contract n^o 33549 for supporting part of his research.

References

J. Allan, J. Aslam, N. Belkin, and C. Buckley. 2002. Challenges in IR and language modeling. In *Report of a Workshop at the University of Amherst*.

X. Carreras and L. Màrquez. 2005. Introduction to the CoNLL-2005 shared task: SRL. In *CoNLL-2005*.

Y. Chen, M. Zhou, and S. Wang. 2006. Reranking answers from definitional QA using language models. In *ACL'06*.

M. Collins and N. Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *ACL'02*.

K. Collins-Thompson, J. Callan, E. Terra, and C. L.A. Clarke. 2004. The effect of document retrieval quality on factoid QA performance. In *SIGIR'04*. ACM.

H. Cui, M. Kan, and T. Chua. 2005. Generic soft pattern models for definitional QA. In *SIGIR'05*. ACM.

T. Joachims. 1999. Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*.

H. Kazawa, H. Isozaki, and E. Maeda. 2001. NTT question answering system in TREC 2001. In *TREC'01*.

P. Kingsbury and M. Palmer. 2002. From Treebank to PropBank. In *LREC'02*.

C. C. T. Kwok, O. Etzioni, and D. S. Weld. 2001. Scaling question answering to the web. In *WWW'01*.

X. Li and D. Roth. 2005. Learning question classifiers: the role of semantic information. *Journ. Nat. Lang. Eng.*

A. Moschitti, B. Coppola, A. Giuglea, and R. Basili. 2005. Hierarchical semantic role labeling. In *CoNLL 2005 shared task*.

A. Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML'06*.

S. Quarteroni and S. Manandhar. 2006. User modelling for Adaptive Question Answering and Information Retrieval. In *FLAIRS'06*.

E. M. Voorhees. 2001. Overview of the TREC 2001 QA track. In *TREC'01*.

D. Zelenko, C. Aone, and A. Richardella. 2003. Kernel methods for relation extraction. *Journ. of Mach. Learn. Res.*

D. Zhang and W. Lee. 2003. Question classification using support vector machines. In *SIGIR'03*. ACM.

Language-independent Probabilistic Answer Ranking for Question Answering

Jeongwoo Ko, Teruko Mitamura, Eric Nyberg
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
{jko, teruko, ehn}@cs.cmu.edu

Abstract

This paper presents a language-independent probabilistic answer ranking framework for question answering. The framework estimates the probability of an individual answer candidate given the degree of answer relevance and the amount of supporting evidence provided in the set of answer candidates for the question. Our approach was evaluated by comparing the candidate answer sets generated by Chinese and Japanese answer extractors with the re-ranked answer sets produced by the answer ranking framework. Empirical results from testing on NTCIR factoid questions show a 40% performance improvement in Chinese answer selection and a 45% improvement in Japanese answer selection.

1 Introduction

Question answering (QA) systems aim at finding precise answers to natural language questions from large document collections. Typical QA systems (Prager et al., 2000; Clarke et al., 2001; Harabagiu et al., 2000) adopt a pipeline architecture that incorporates four major steps: (1) question analysis, (2) document retrieval, (3) answer extraction and (4) answer selection. Question analysis is a process which analyzes a question and produces a list of keywords. Document retrieval is a step that searches for relevant documents or passages. Answer extraction extracts a list of answer candidates from the retrieved documents. Answer selection is a process which pinpoints correct answer(s) from the extracted candidate answers.

Since the first three steps in the QA pipeline may produce erroneous outputs, the final answer selection step often entails identifying correct answer(s) amongst many incorrect ones. For example, given the question *"Which Chinese city has the largest number of foreign financial companies?"*, the answer extraction component produces a ranked list of five answer candidates: Beijing (AP880603-0268)[1], Hong Kong (WSJ920110-0013), Shanghai (FBIS3-58), Taiwan (FT942-2016) and Shanghai (FBIS3-45320). Due to imprecision in answer extraction, an incorrect answer ("Beijing") can be ranked in the first position, and the correct answer ("Shanghai") was extracted from two different documents and ranked in the third and the fifth positions. In order to rank "Shanghai" in the top position, we have to address two interesting challenges:

- *Answer Similarity*. How do we exploit similarity among answer candidates? For example, when the candidates list contains redundant answers (e.g., "Shanghai" as above) or several answers which represent a single instance (e.g. "U.S.A." and "the United States"), how much should we boost the rank of the redundant answers?

- *Answer Relevance*. How do we identify relevant answer(s) amongst irrelevant ones? This task may involve searching for evidence of a relationship between the answer

[1] Answer candidates are shown with the identifier of the TREC document where they were found.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 784–791,
Prague, Czech Republic, June 2007.

and the answer type or a question keyword. For example, we might wish to query a knowledge base to determine if "Shanghai" is a city (`IS-A(Shanghai, city)`), or to determine if Shanghai is in China (`IS-IN(Shanghai, China)`).

The first challenge is to exploit redundancy in the set of answer candidates. As answer candidates are extracted from different documents, they may contain identical, similar or complementary text snippets. For example, "U.S." can appear as "United States" or "USA" in different documents. It is important to detect redundant information and boost answer confidence, especially for list questions that require a set of unique answers. One approach is to perform answer clustering (Nyberg et al., 2002; Jijkoun et al., 2006). However, the use of clustering raises additional questions: how to calculate the score of the clustered answers, and how to select the cluster label.

To address the second question, several answer selection approaches have used external knowledge resources such as WordNet, CYC and gazetteers for answer validation or answer reranking. Answer candidates are either removed or discounted if they are not of the expected answer type (Xu et al., 2002; Moldovan et al., 2003; Chu-Carroll et al., 2003; Echihabi et al., 2004). The Web also has been used for answer reranking by exploiting search engine results produced by queries containing the answer candidate and question keywords (Magnini et al., 2002). This approach has been used in various languages for answer validation. Wikipedia's structured information was used for Spanish answer type checking (Buscaldi and Rosso, 2006).

Although many QA systems have incorporated individual features and/or resources for answer selection in a single language, there has been little research on a generalized probabilistic framework that supports answer ranking in multiple languages using any answer relevance and answer similarity features that are appropriate for the language in question.

In this paper, we describe a probabilistic answer ranking framework for multiple languages. The framework uses logistic regression to estimate the probability that an answer candidate is correct given multiple answer relevance features and answer similarity features. An existing framework which was originally developed for English (Ko et al., 2007) was extended for Chinese and Japanese answer ranking by incorporating language-specific features. Empirical results on NTCIR Chinese and Japanese factoid questions show that the framework significantly improved answer selection performance; Chinese performance improved by 40% over the baseline, and Japanese performance improved by 45% over the baseline.

The remainder of this paper is organized as follows: Section 2 contains an overview of the answer ranking task. Section 3 summarizes the answer ranking framework. In Section 4, we explain how we extended the framework by incorporating language-specific features. Section 5 describes the experimental methodology and results. Finally, Section 6 concludes with suggestions for future research.

2 Answer Ranking Task

The relevance of an answer to a question can be estimated by the probability P($correct(A_i)$ $|A_i, Q$), where Q is a question and A_i is an answer candidate. To exploit answer similarity, we estimate the probability $P(correct(A_i)\ |A_i, A_j)$, where A_j is similar to A_i. Since both probabilities influence overall answer ranking performance, it is important to combine them in a unified framework and estimate the probability of an answer candidate as: $P(correct(A_i)|Q, A_1, ..., A_n)$.

The estimated probability is used to rank answer candidates and select final answers from the list. For factoid questions, the top answer is selected as a final answer to the question. In addition, we can use the estimated probability to classify incorrect answers: if the probability of an answer candidate is lower than 0.5, it is considered to be a wrong answer and is filtered out of the answer list. This is useful in deciding whether or not a valid answer to a question exists in a given corpus (Voorhees, 2002). The estimated probability can also be used in conjunction with a cutoff threshold when selecting multiple answers to list questions.

3 Answer Ranking Framework

This section summarizes our answer ranking framework, originally developed for English answers (Ko

$$
\begin{aligned}
&P(correct(A_i)|Q, A_1, ..., A_n) \\
&\quad \approx P(correct(A_i)|rel_1(A_i), ..., rel_{K1}(A_i), sim_1(A_i), ..., sim_{K2}(A_i)) \\
&\quad = \frac{exp(\alpha_0 + \sum_{k=1}^{K1} \beta_k rel_k(A_i) + \sum_{k=1}^{K2} \lambda_k sim_k(A_i))}{1 + exp(\alpha_0 + \sum_{k=1}^{K1} \beta_k rel_k(A_i) + \sum_{k=1}^{K2} \lambda_k sim_k(A_i))} \\
&\quad where, sim_k(A_i) = \sum_{j=1(j \neq i)}^{N} sim'_k(A_i, A_j).
\end{aligned}
$$

Figure 1: Estimating correctness of an answer candidate given a question and a set of answer candidates

et al., 2007). The model uses logistic regression to estimate the probability of an answer candidate (Figure 1). Each $\text{rel}_k(\text{A}_i)$ is a feature function used to produce an answer relevance score for an answer candidate A_i. Each $\text{sim}'_k(A_i, A_j)$ is a similarity function used to calculate an answer similarity between A_i and A_j. K1 and K2 are the number of answer relevance and answer similarity features, respectively. N is the number of answer candidates.

To incorporate multiple similarity features, each $\text{sim}_k(A_i)$ is obtained from an individual similarity metric, $\text{sim}'_k(A_i, A_j)$. For example, if Levenshtein distance is used as one similarity metric, $\text{sim}_k(A_i)$ is calculated by summing N-1 Levenshtein distances between one answer candidate and all other candidates.

The parameters α, β, λ were estimated from training data by maximizing the log likelihood. We used the Quasi-Newton algorithm (Minka, 2003) for parameter estimation.

Multiple features were used to generate answer relevance scores and answer similarity scores; these are discussed below.

3.1 Answer Relevance Features

Answer relevance features can be classified into knowledge-based features or data-driven features.

1) Knowledge-based features

Gazetteers: Gazetteers provide geographic information, which allows us to identify strings as instances of countries, their cities, continents, capitals, etc. For answer ranking, three gazetteer resources were used: the Tipster Gazetteer, the CIA World Factbook and information about the US states provided by 50states.com. These resources were used to assign an answer relevance score between -1 and 1 to each candidate. For example, given the question "Which city in China has the largest number of foreign financial companies?", the candidate "Shanghai" receives a score of 0.5 because it is a city in the gazetteers. But "Taiwan" receives a score of -1.0 because it is not a city in the gazetteers. A score of 0 means the gazetteers did not contribute to the answer selection process for that candidate.

Ontology: Ontologies such as WordNet contain information about relationships between words and general meaning types (synsets, semantic categories, etc.). WordNet was used to identify answer relevance in a manner analogous to the use of gazetteers. For example, given the question "Who wrote the book 'Song of Solomon'?", the candidate "Mark Twain" receives a score of 0.5 because its hypernyms include "writer".

2) Data-driven features

Wikipedia: Wikipedia was used to generate an answer relevance score. If there is a Wikipedia document whose title matches an answer candidate, the document is analyzed to obtain the term frequency (tf) and the inverse term frequency (idf) of the candidate, from which a tf.idf score is calculated. When there is no matched document, each question keyword is also processed as a back-off strategy, and the answer relevance score is calculated by summing the tf.idf scores obtained from individual keywords.

Google: Following Magnini et al. (2002), a query consisting of an answer candidate and question key-

words was sent to the Google search engine. Then the top 10 text snippets returned by Google were analyzed to generate an answer relevance score by computing the minimum number of words between a keyword and the answer candidate.

3.2 Answer Similarity Features

Answer similarity is calculated using multiple string distance metrics and a list of synonyms.

String Distance Metrics: String distance metrics such as Levenshtein, Jaro-Winkler, and Cosine similarity were used to calculate the similarity between two English answer candidates.

Synonyms: Synonyms can be used as another metric to calculate answer similarity. If one answer is synonym of another answer, the score is 1. Otherwise the score is 0. To get a list of synonyms, three knowledge bases were used: WordNet, Wikipedia and the CIA World Factbook. In addition, manually generated rules were used to obtain synonyms for different types of answer candidates. For example, "April 12 1914" and "12th Apr. 1914" are converted into "1914-04-12" and treated as synonyms.

4 Extensions for Multiple Languages

We extended the framework for Chinese and Japanese QA. This section details how we incorporated language-specific resources into the framework. As logistic regression is based on a probabilistic framework, the model does not need to be changed to support other languages. We only retrained the model for individual languages. To support Chinese and Japanese QA, we incorporated new features for individual languages.

4.1 Answer Relevance Features

We replaced the English gazetteers and WordNet with language-specific resources for Japanese and Chinese. As Wikipedia and the Web support multiple languages, the same algorithm was used in searching language-specific corpora for the two languages.

1) Knowledge-based features

The knowledge-based features involve searching for facts in a knowledge base such as gazetteers and WordNet. We utilized comparable resources for Chinese and Japanese. Using language-specific resources, the same algorithms were applied to generate an answer relevance score between -1 and 1.

Language	#Articles	
	Nov. 2005	Aug. 2006
English	1,811,554	3,583,699
Japanese	201,703	446,122
Chinese	69,936	197,447

Table 1: Articles in Wikipedia for different languages

Gazetteers: There are few available gazetteers for Chinese and Japanese. Therefore, we extracted location data from language-specific resources. For Japanese, we extracted Japanese location information from Yahoo[2], which contains many location names in Japan and the relationships among them. For Chinese, we extracted location names from the Web. In addition, we translated country names provided by the CIA World Factbook and the Tipster gazetteers into Chinese and Japanese names. As there is more than one translation, top 3 translations were used.

Ontology: For Chinese, we used HowNet (Dong, 2000) which is a Chinese version of WordNet. It contains 65,000 Chinese concepts and 75,000 corresponding English equivalents. For Japanese, we used semantic classes provided by Gengo GoiTaikei[3]. Gengo GoiTaikei is a Japanese lexicon containing 300,000 Japanese words with their associated 3,000 semantic classes. The semantic information provided by HowNet and Gengo GoiTaikei was used to assign an answer relevance score between -1 and 1.

2) Data-driven features

Wikipedia: As Wikipedia supports more than 200 language editions, the approach used in English can be used for different languages without any modification. Table 1 shows the number of text articles in three different languages. Wikipedia's current coverage in Japanese and Chinese does not match its coverage in English, but coverage in these languages continues to improve.

To supplement the small corpus of Chinese documents available, we used Baidu

[2]http://map.yahoo.co.jp/

[3]http://www.kecl.ntt.co.jp/mtg/resources/GoiTaikei

(http://baike.baidu.com), which is similar to Wikipedia but contains more articles written in Chinese. We first search for Chinese Wikipedia. When there is no matching document in Wikipedia, each answer candidate is sent to Baidu and the retrieved document is analyzed in the same way to analyze Wikipedia documents.

The idf score was calculated using word statistics from Japanese Yomiuri newspaper corpus and the NTCIR Chinese corpus.

Google: The same algorithm was applied to analyze Japanese and Chinese snippets returned from Google. But we restricted the language to Chinese or Japanese so that Google returned only Chinese or Japanese documents. To calculate the distance between an answer candidate and question keywords, segmentation was done with linguistic tools. For Japanese, Chasen[4] was used. For Chinese segmentation, a maximum-entropy based parser was used (Wang et al., 2006).

3) Manual Filtering

Other than the features mentioned above, we manually created many rules for numeric and temporal questions to filter out invalid answers. For example, when the question is looking for a year as an answer, an answer candidate which contains only the month receives a score of -1. Otherwise, the score is 0.

4.2 Answer Similarity Features

The same features used for English were applied to calculate the similarity of Chinese/Japanese answer candidates. To identify synonyms, Wikipedia were used for both Chinese and Japanese. EIJIRO dictionary was used to obtain Japanese synonyms. EIJIRO is a English-Japanese dictionary containing 1,576,138 words and provides synonyms for Japanese words.

As there are several different ways to represent temporal and numeric expressions (Nyberg et al., 2002; Greenwood, 2006), language-specific conversion rules were applied to convert them into a canonical format; for example, a rule to convert Japanese Kanji characters to Arabic numbers is shown in Figure 2.

[4]http://chasen.aist-nara.ac.jp/hiki/ChaSen

Original answer string	Normalized answer string
三 千 億 円	3E+11 円
3,000億円	3E+11 円
一九九三年 七月 四 日	1993-07-04
1993 年 7 月 4 日	1993-07-04
四分の一	0.25
５割	50 %

Figure 2: Example of normalized answer strings

5 Experiments

This section describes the experiments to evaluate the extended answer ranking framework for Chinese and Japanese QA.

5.1 Experimental Setup

We used Chinese and Japanese questions provided by the NTCIR (NII Test Collection for IR Systems), which focuses on evaluating cross-lingual and monolingual QA tasks for Chinese, Japanese and English. For Chinese, a total of 550 factoid questions from the NTCIR5-6 QA evaluations served as the dataset. Among them, 200 questions were used to train the Chinese answer extractor and 350 questions were used to evaluate our answer ranking framework. For Japanese, 700 questions from the NTCIR5-6 QA evaluations served as the dataset. Among them, 300 questions were used to train the Japanese answer extractor and 400 questions were used to evaluate our framework.

Both the Chinese and Japanese answer extractors use maximum-entropy to extract answer candidates based on multiple features such as named entity, dependency structures and some language-dependent features.

Performance of the answer ranking framework was measured by average answer accuracy: the number of correct top answers divided by the number of questions where at least one correct answer exists in the candidate list provided by an extractor. Mean Reciprocal Rank (MRR5) was also used to calculate the average reciprocal rank of the first correct answer in the top 5 answers.

The baseline for average answer accuracy was calculated using the answer candidate likelihood scores provided by each individual extractor; the

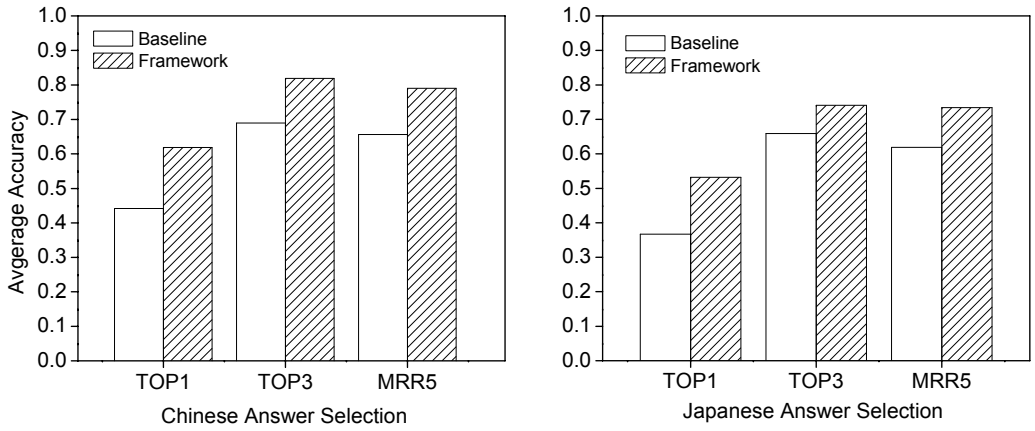


Figure 3: Performance of the answer ranking framework for Chinese and Japanese answer selection (TOP1: average accuracy of top answer, TOP3: average accuracy of top 3 answers, MRR5: average of mean reciprocal rank of top 5 answers)

answer with the best extractor score was chosen, and no validation or similarity processing was performed.

3-fold cross-validation was performed, and we used a version of Wikipedia downloaded in Aug 2006.

5.2 Results and Analysis

We first analyzed the average accuracy of top 1, top3 and top 5 answers. Figure 3 compares the average accuracy using the baseline and the answer selection framework. As can be seen, the answer ranking framework significantly improved performance on both Chinese and Japanese answer selection. As for the average top answer accuracy, there were 40% improvement over the baseline (Chinese) and 45% improvement over the baseline (Japanese).

We also analyzed the degree to which the average accuracy was affected by answer similarity and relevance features. Table 2 compares the average top answer accuracy using the baseline, the answer relevance features, the answer similarity features and all feature combinations. Both the similarity and the relevance features significantly improved answer selection performance compared to the baseline, and combining both sets of features together produced the best performance.

We further analyzed the utility of individual relevance features (Figure 4). For both languages, filtering was useful in ruling out wrong answers. The impact of the ontology was more positive for Japanese; we assume that this is because the Chinese ontology (HowNet) contains much less information overall than the Japanese ontology (Gengo GoiTaikei). The comparative impact of Wikipedia was similar. For Chinese, there were many fewer Wikipedia documents available. Even though we used Baidu as a supplemental resource for Chinese, this did not improve answer selection performance. On the other hand, the use of Wikipedia was very helpful for Japanese, improving performance by 26% over the baseline. This shows that the quality of answer relevance estimation is significantly affected by resource coverage.

	Baseline	Rel	Sim	All
Chinese	0.442	0.482	0.597	0.619
Japanese	0.367	0.463	0.502	0.532

Table 2: Average top answer accuracy of individual features (Rel: merging relevance features, Sim: merging similarity features, ALL: merging all features).

When comparing the data-driven features with the knowledge-based features, the data-driven features (such as Wikipedia and Google) tended to increase performance more than the knowledge-based features (such as gazetteers and WordNet).

Table 3 shows the effect of individual similarity features on Chinese and Japanese answer selec-

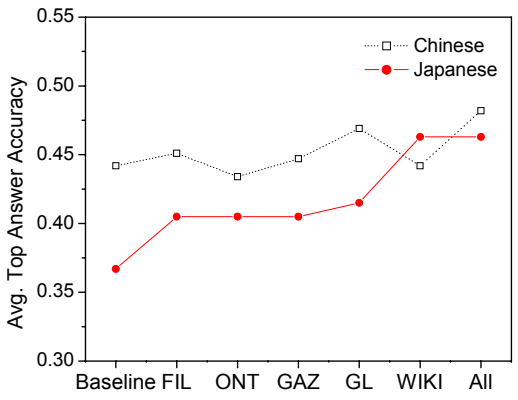


Figure 4: Average top answer accuracy of individual answer relevance features.(FIL: filtering, ONT, ontology, GAZ: gazetteers, GL: Google, WIKI: Wikipedia, ALL: combination of all relevance features)

	Chinese		Japanese	
	0.3	0.5	0.3	0.5
Cosine	**0.597**	**0.597**	0.488	0.488
Jaro-Winkler	0.544	0.518	0.410	0.415
Levenshtein	0.558	0.544	0.434	0.449
Synonyms	0.527	0.527	0.493	0.493
All	0.588	0.580	**0.502**	0.488

Table 3: Average accuracy using individual similarity features under different thresholds: 0.3 and 0.5 ("All": combination of all similarity metrics)

tion. As some string similarity features (e.g., Levenshtein distance) produce a number between 0 and 1 (where 1 means two strings are identical and 0 means they are different), similarity scores less than a threshold can be ignored. We used two thresholds: 0.3 and 0.5. In our experiments, using 0.3 as a threshold produced better results in Chinese. In Japanese, 0,5 was a better threshold for individual features. Among three different string similarity features (Levenshtein, Jaro-Winkler and Cosine similarity), cosine similarity tended to perform better than the others.

When comparing synonym features with string similarity features, synonyms performed better than string similarity in Japanese, but not in Chinese. We had many more synonyms available for Japanese and they helped the system to better exploit answer redundancy.

	Data-driven features	All features
Chinese	0.606	0.619
Japanese	0.517	0.532

Table 4: Average top answer accuracy when using data-driven features v.s. when using all features.

We also analyzed answer selection performance when combining all four similarity features ("All" in Table 3). Combining all similarity features improved the performance in Japanese, but hurt the performance in Chinese, because adding a small set of synonyms to the string metrics worsened the performance of logistic regression.

5.3 Utility of data-driven features

In our experiments we used data-driven features as well as knowledge-based features. As knowledge-based features need manual effort to access language-specific resources for individual languages, we conducted an additional experiment only with data-driven features in order to see how much performance gain is available without the manual work. As Google, Wikipedia and string similarity metrics can be used without any additional manual effort when extended to other languages, we used these three features and compared the performance.

Table 4 shows the performance when using data-driven features v.s. all features. It can be seen that data-driven features alone achieved significant improvement over the baseline. This indicates that the framework can easily be extended to any language where appropriate data resources are available, even if knowledge-based features and resources for the language are still under development.

6 Conclusion

In this paper, we presented a generalized answer selection framework which was applied to Chinese and Japanese question answering. An empirical evaluation using NTCIR test questions showed that the framework significantly improves baseline answer selection performance. For Chinese, the performance improved by 40% over the baseline. For Japanese, the performance improved by 45% over

the baseline. This shows that our probabilistic framework can be easily extended for multiple languages by reusing data-driven features (with new corpora) and adding language-specific resources (ontologies, gazetteers) for knowledge-based features.

In our previous work, we evaluated the performance of the framework for English QA using questions from past TREC evaluations (Ko et al., 2007). The experimental results showed that the combination of all answer ranking features improved performance by an average of 102% over the baseline. The relevance features improved performance by an average of 99% over the baseline, and the similarity features improved performance by an average of 46% over the baseline. Our hypothesis is that answer relevance features had a greater impact for English QA because the quality and coverage of the data resources available for English answer validation is much higher than the quality and coverage of existing resources for Japanese and Chinese. In future work, we will continue to evaluate the robustness of the framework. It is also clear from our comparison with English QA that more work can and should be done in acquiring data resources for answer validation in Chinese and Japanese.

Acknowledgments

We would like to thank Hideki Shima, Mengqiu Wang, Frank Lin, Justin Betteridge, Matthew Bilotti, Andrew Schlaikjer and Luo Si for their valuable support. This work was supported in part by ARDA/DTO AQUAINT program award number NBCHC040164.

References

D. Buscaldi and P. Rosso. 2006. Mining Knowledge from Wikipedia for the Question Answering task. In *Proceedings of the International Conference on Language Resources and Evaluation*.

J. Chu-Carroll, J. Prager, C. Welty, K. Czuba, and D. Ferrucci. 2003. A Multi-Strategy and Multi-Source Approach to Question Answering. In *Proceedings of Text REtrieval Conference*.

C. Clarke, G. Cormack, and T. Lynam. 2001. Exploiting redundancy in question answering. In *Proceedings of SIGIR*.

Zhendong Dong. 2000. Hownet: http://www.keenage.com.

A. Echihabi, U. Hermjakob, E. Hovy, D. Marcu, E. Melz, and D. Ravichandran. 2004. How to select an answer string? In T. Strzalkowski and S. Harabagiu, editors, *Advances in Textual Question Answering*. Kluwer.

Mark A. Greenwood. 2006. Open-Domain Question Answering. Thesis.

S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunsecu, R. Girju, V. Rus, and P. Morarescu. 2000. Falcon: Boosting knowledge for answer engines. In *Proceedings of TREC*.

V. Jijkoun, J. van Rantwijk, D. Ahn, E. Tjong Kim Sang, and M. de Rijke. 2006. The University of Amsterdam at CLEF@QA 2006. In *Working Notes CLEF*.

J. Ko, L. Si, and E. Nyberg. 2007. A Probabilistic Framework for Answer Selection in Question Answering. In *Proceedings of NAACL/HLT*.

B. Magnini, M. Negri, R. Pervete, and H. Tanev. 2002. Comparing statistical and content-based techniques for answer validation on the web. In *Proceedings of the VIII Convegno AI*IA*.

T. Minka. 2003. A Comparison of Numerical Optimizers for Logistic Regression. Unpublished draft.

D. Moldovan, D. Clark, S. Harabagiu, and S. Maiorano. 2003. Cogex: A logic prover for question answering. In *Proceedings of HLT-NAACL*.

E. Nyberg, T. Mitamura, J. Carbonell, J. Callan, K. Collins-Thompson, K. Czuba, M. Duggan, L. Hiyakumoto, N. Hu, Y. Huang, J. Ko, L. Lita, S. Murtagh, V. Pedro, and D. Svoboda. 2002. The JAVELIN Question-Answering System at TREC 2002. In *Proceedings of Text REtrieval Conference*.

J. Prager, E. Brown, A. Coden, and D. Radev. 2000. Question answering by predictive annotation. In *Proceedings of SIGIR*.

E. Voorhees. 2002. Overview of the TREC 2002 question answering track. In *Proceedings of Text REtrieval Conference*.

M. Wang, K. Sagae, and T. Mitamura. 2006. A Fast, Accurate Deterministic Parser for Chinese. In *Proceedings of COLING/ACL*.

J. Xu, A. Licuanan, J. May, S. Miller, and R. Weischedel. 2002. TREC 2002 QA at BBN: Answer Selection and Confidence Estimation. In *Proceedings of Text REtrieval Conference*.

Learning to Compose Effective Strategies from a Library of Dialogue Components

Martijn Spitters[†] **Marco De Boni**[‡] **Jakub Zavrel**[†] **Remko Bonnema**[†]

[†] Textkernel BV, Nieuwendammerkade 28/a17, 1022 AB Amsterdam, NL
{spitters,zavrel,bonnema}@textkernel.nl
[‡] Unilever Corporate Research, Colworth House, Sharnbrook, Bedford, UK MK44 1LQ
marco.de-boni@unilever.com

Abstract

This paper describes a method for automatically learning effective dialogue strategies, generated from a library of dialogue content, using reinforcement learning from user feedback. This library includes greetings, social dialogue, chit-chat, jokes and relationship building, as well as the more usual clarification and verification components of dialogue. We tested the method through a motivational dialogue system that encourages take-up of exercise and show that it can be used to construct good dialogue strategies with little effort.

1 Introduction

Interactions between humans and machines have become quite common in our daily life. Many services that used to be performed by humans have been automated by natural language dialogue systems, including information seeking functions, as in timetable or banking applications, but also more complex areas such as tutoring, health coaching and sales where communication is much richer, embedding the provision and gathering of information in e.g. social dialogue. In the latter category of dialogue systems, a high level of naturalness of interaction and the occurrence of longer periods of satisfactory engagement with the system are a prerequisite for task completion and user satisfaction.

Typically, such systems are based on a dialogue strategy that is manually designed by an expert based on knowledge of the system and the domain, and on continuous experimentation with test users. In this process, the expert has to make many design choices which influence task completion and user satisfaction in a manner which is hard to assess, because the effectiveness of a strategy depends on many different factors, such as classification/ASR performance, the dialogue domain and task, and, perhaps most importantly, personality characteristics and knowledge of the user.

We believe that the key to maximum dialogue effectiveness is to listen to the user. This paper describes the development of an adaptive dialogue system that uses the feedback of users to automatically improve its strategy. The system starts with a library of generic and task-/domain-specific dialogue components, including social dialogue, chit-chat, entertaining parts, profiling questions, and informative and diagnostic parts. Given this variety of possible dialogue actions, the system can follow many different strategies within the dialogue state space. We conducted training sessions in which users interacted with a version of the system which randomly generates a possible dialogue strategy for each interaction (restricted by global dialogue constraints). After each interaction, the users were asked to reward different aspects of the conversation. We applied reinforcement learning to use this feedback to compute the optimal dialogue policy.

The following section provides a brief overview of previous research related to this area and how our work differs from these studies. We then proceed with a concise description of the dialogue system used for our experiments in section 3. Section 4 is about the training process and the reward model. Section 5 goes into detail about dialogue policy op-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 792–799,
Prague, Czech Republic, June 2007.

timization with reinforcement learning. In section 6 we discuss our experimental results.

2 Related Work

Previous work has examined learning of effective dialogue strategies for information seeking spoken dialogue systems, and in particular the use of reinforcement learning methods to learn policies for action selection in dialogue management (see e.g. Levin et al., 2000; Walker, 2000; Scheffler and Young, 2002; Peek and Chickering, 2005; Frampton and Lemon, 2006), for selecting initiative and confirmation strategies (Singh et al., 2002); for detecting speech recognition problem (Litman and Pan, 2002); changing the dialogue according to the expertise of the user (Maloor and Chai, 2000); adapting responses according to previous interactions with the users (Rudary et al., 2004); optimizing mixed initiative in collaborative dialogue (English and Heeman, 2005), and optimizing confirmations (Cuayáhuitl et al., 2006). Other researchers have focussed their attention on the learning aspect of the task, examining, for example hybrid reinforcement/supervised learning (Henderson et al., 2005).

Previous work on learning dialogue management strategies has however generally been limited to well defined areas of the dialogue, in particular dealing with speech recognition and clarification problems, with small state spaces and a limited set of actions to choose from (Henderson et al., 2005). In a number of contexts, however, dialogues need to have a far greater degree of complexity not just in the number of states and possible actions but also in the variety of dialogue acts: for example in motivational dialogue systems where the task is not limited to information gathering, slot-filling or querying of a database, and where dialogues must contain more social and relational elements to be successful (for the usefulness of social dialogue see e.g. Bickmore, 2003; Liu and Picard, 2005). Only little effort has been directed to the question what dialogue components should make up the dialogue, involving decisions like how much and what type of social interaction should be used, different ways of forming a relationship with the user such as using chit-chat (for example asking about a user's hobbies or asking for the user's name), using humour, as well as the more conventional tasks of clarifying user input, establishing common ground and ensuring system replies are appropriate. Our work has focused on these aspects of dialogue strategy construction, in order to create good dialogue strategies incorporating appropriate levels of social interaction, humour, chit-chat, as well as successful information gathering and provision.

3 A Motivational Dialogue System

The domain of our system is physical exercise. The system is set up as an exercise advisor that asks the user what is preventing him/her from exercising more. After the user has worded his/her exercise 'barrier', the system will give motivational advice for how to overcome this barrier. As an illustration, Table 1 shows an example dialogue, generated by our system. Our dialogue system is *text*-based, so it does not involve speech recognition. This section describes the system's three main modules: the dialogue state/action manager, the utterance classifier and the dialogue component library.

Speaker	Utterance
system	*Good evening!*
system	*What's your name?*
user	**my name is Brenda**
system	*So Brenda, how are you today?*
user	**fine thank you**
system	*Tell me what is preventing you from exercising more.*
user	**I don't have anyone to exercise with...**
system	*So your barrier is Loneliness (e.g. You don't like exercising on your own).*
user	**sort of, yeah**
system	*A solution for this barrier is: Try going on your own and see whether it's really that different to going with a friend. Try talking to other people who are exercising, you're likely to find they welcome a little conversation!*
system	*Did you find this solution useful?*
user	**I guess...**
system	*Ok. Try to work on this to overcome your barrier.*
system	*Bye!*

Table 1: Example dialogue.

3.1 Dialogue state/action management

The dialogue state is unique at every stage of the conversation and is represented as a vector of feature-values. We use only a limited set of features because, as also noted in (Singh et al., 2002; Levin et al., 2000), it is important to keep the state space as small as possible (but with enough distinc-

tive power to support learning) so we can construct a non-sparse Markov decision process (see section 5) based on our limited training dialogues. The state features are listed in Table 2.

Feature	Values	Description
curnode	$c \in N$	the current dialogue node
actiontype	$utt, trans$	action type
trigger	$t \in T$	utterance classifier category
confidence	$1, 0$	category confidence
problem	$1, 0$	communication problem earlier

Table 2: Dialogue state features.

In each dialogue state, the dialogue manager will look up the next action that should be taken. In our system, an action is either a *system utterance* or a *transition* in the dialogue structure. In the initial system, the dialogue structure was manually constructed. In many states, the next action requires a choice to be made. Dialogue states in which the system can choose among several possible actions are called *choice-states*. For example, in our system, immediately after greeting the user, the dialogue structure allows for different directions: the system can first ask some personal questions, or it can immediately discuss the main topic without any digressions. Utterance actions may also require a choice (e.g. *directive* versus *open* formulation of a question). In training mode, the system will make random choices in the choice-states. This approach will generate many different dialogue strategies, i.e. *paths* through the dialogue structure.

User replies are sent to an utterance classifier. The category assigned by this classifier is returned to the dialogue manager and triggers a transition to the next node in the dialogue structure. The system also accommodates a simple rule-based extraction module, which can be used to extract information from user utterances (e.g. the user's name, which is templated in subsequent system prompts in order to personalize the dialogue).

3.2 Utterance classification

The (memory-based) classifier uses a rich set of features for accurate classification, including words, phrases, regular expressions, domain-specific word-relations (using a taxonomy-plugin) and syntactically motivated expressions. For utterance parsing we used a memory-based shallow parser, called MBSP (Daelemans et al., 1999). This parser provides part of speech labels, chunk brackets, subject-verb-object relations, and has been enriched with detection of negation scope and clause boundaries.

The feature-matching mechanism in our classification system can match terms or phrases at specified positions in the token stream of the utterance, also in combination with syntactic and semantic class labels. This allows us to define features that are particularly useful for resolving confusing linguistic phenomena like ambiguity and negation. A base feature set was generated automatically, but quite a lot of features were manually tuned or added to cope with certain common dialogue situations. The overall classification accuracy, measured on the dialogues that were produced during the training phase, is 93.6%. Average precision/recall is 98.6/97.3% for the non-barrier categories (*confirmation*, *negation*, *unwillingness*, etc.), and 99.1/83.4% for the barrier categories (*injury*, *lack of motivation*, etc.).

3.3 Dialogue Component Library

The dialogue component library contains generic as well as task-/domain-specific dialogue content, combining different aspects of dialogue (task/topic structure, communication goals, etc.). Table 3 lists all components in the library that was used for training our dialogue system. A dialogue component is basically a coherent set of dialogue node representations with a certain dialogue function. The library is set up in a flexible, generic way: new components can easily be plugged in to test their usefulness in different dialogue contexts or for new domains.

4 Training the Dialogue System

4.1 Random strategy generation

In its training mode, the dialogue system uses random exploration: it generates different dialogue strategies by choosing randomly among the *allowed actions* in the choice-states. Note that dialogue generation is constrained to contain certain fixed actions that are essential for task completion (e.g. asking the exercise barrier, giving a solution, closing the session). This excludes a vast number of useless strategies from exploration by the system. Still, given all action choices and possible user reactions, the total number of unique dialogues that can be generated by

Component	Description	p_a	p_e
StartSession	Dialogue openings, including various greetings	•	•
PersonalQuestionnaire	Personal questions, e.g. name; age; hobbies; interests, *how are you today?*	•	
ElizaChitChat	Eliza-like chit-chat, e.g. *please go on...*		
ExerciseChitChat	Chit-chat about exercise, e.g. *have you been doing any exercise this week?*	○	
Barrier	Prompts concerning the barrier, e.g. ask the barrier; barrier verification; ask a rephrase	•	•
Solution	Prompts concerning the solution, e.g. give the solution; verify usefulness	•	•
GiveBenefits	Talk about the benefits of exercising		
AskCommitment	Ask user to commit his implementation of the given solution	•	
Encourage	Encourage the user to work on the given solution	•	•
GiveJoke	The humor component: ask if the user wants to hear a joke; tell random jokes	○	•
VerifyCloseSession	Verification for closing the session (*are you sure you want to close this session?*)	○	○
CloseSession	Dialogue endings, including various farewells	•	•

Table 3: Components in the dialogue component library. The last two columns show which of the components was used in the learned policy (p_a) and the expert policy (p_e), discussed in section 6. • means the component is always used, ○ means it is sometimes used, depending on the dialogue state.

the system is approximately 345000 (many of which are unlikely to ever occur). During training, the system generated 490 different strategies. There are 71 choice-states that can actually occur in a dialogue. In our training dialogues, the opening state was obviously visited most frequently (572 times), almost 60% of all states was visited at least 50 times, and only 16 states were visited less than 10 times.

4.2 The reward model

When the dialogue has reached its final state, a survey is presented to the user for dialogue evaluation. The survey consists of five statements that can each be rated on a five-point scale (indicating the user's level of agreement). The responses are mapped to rewards of -2 to 2. The statements we used are partly based on the user survey that was used in (Singh et al., 2002). We considered these statements to reflect the most important aspects of conversation that are relevant for learning a good dialogue policy. The five statements we used are listed below.

M1 *Overall, this conversation went well*

M2 *The system understood what I said*

M3 *I knew what I could say at each point in the dialogue*

M4 *I found this conversation engaging*

M5 *The system provided useful advice*

4.3 Training set-up

Eight subjects carried out a total of 572 conversations with the system. Because of the variety of possible exercise barriers known by the system (52 in total) and the fact that some of these barriers are more complex or harder to detect than others, the system's classification accuracy depends largely on the user's barrier. To prevent classification accuracy distorting the user evaluations, we asked the subjects to act as if they had one of five predefined exercise barriers (e.g. *Imagine that you don't feel comfortable exercising in public. See what the advisor recommends for this barrier to your exercise*).

5 Dialogue Policy Optimization with Reinforcement Learning

Reinforcement learning refers to a class of machine learning algorithms in which an agent explores an environment and takes actions based on its current state. In certain states, the environment provides a reward. Reinforcement learning algorithms attempt to find the optimal policy, i.e. the policy that maximizes cumulative reward for the agent over the course of the problem. In our case, a policy can be seen as a mapping from the dialogue states to the possible actions in those states. The environment is typically formulated as a Markov decision process (MDP).

The idea of using reinforcement learning to automate the design of strategies for dialogue systems was first proposed by Levin et al. (2000) and has subsequently been applied in a.o. (Walker, 2000; Singh et al., 2002; Frampton and Lemon, 2006; Williams et al., 2005).

5.1 Markov decision processes

We follow past lines of research (such as Levin et al., 2000; Singh et al., 2002) by representing a dialogue as a trajectory in the state space, determined

by the user responses and system actions: $s_1 \xrightarrow{a_1,r_1} s_2 \xrightarrow{a_2,r_2} \ldots s_n \xrightarrow{a_n,r_n} s_{n+1}$, in which $s_i \xrightarrow{a_i,r_i} s_{i+1}$ means that the system performed action a_i in state s_i, received[1] reward r_i and changed to state s_{i+1}. In our system, a state is a dialogue context vector of feature values. This feature vector contains the available information about the dialogue so far that is relevant for deciding what action to take next in the current dialogue state. We want the system to learn the optimal decisions, i.e. to choose the actions that maximize the expected reward.

5.2 Q-value iteration

The field of reinforcement learning includes many algorithms for finding the optimal policy in an MDP (see Sutton and Barto, 1998). We applied the algorithm of (Singh et al., 2002), as their experimental set-up is similar to ours, constisting of: generation of (limited) exploratory dialogue data, using a training system; creating an MDP from these data and the rewards assigned by the training users; off-line policy learning based on this MDP.

The Q-function for a certain action taken in a certain state describes the total reward expected between taking that action and the end of the dialogue. For each state-action pair (s, a), we calculated this expected cumulative reward $Q(s, a)$ of taking action a from state s, with the following equation (Sutton and Barto, 1998; Singh et al., 2002):

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a') \quad (1)$$

where: $P(s'|s,a)$ is the probability of a transition from state s to state s' by taking action a, and $R(s,a)$ is the expected reward obtained when taking action a in state s. γ is a weight ($0 \leq \gamma \leq 1$), that discounts rewards obtained later in time when it is set to a value < 1. In our system, γ was set to 1. Equation 1 is recursive: the Q-value of a certain state is computed in terms of the Q-values of its successor states. The Q-values can be estimated to within a desired threshold using Q-value iteration (Sutton and Barto, 1998). Once the value iteration process is completed, by selecting the action with the maximum Q-value (the maximum expected future reward) at each choice-state, we can obtain the optimal dialogue policy π.

[1]In our experiments, we did not make use of immediate rewarding (e.g. at every turn) during the conversation. Rewards were given after the final state of the dialogue had been reached.

6 Results and Discussion

6.1 Reward analysis

Figure 1 shows a graph of the distribution of the five different evaluation measures in the training data (see section 4.2 for the statement wordings). M1 is probably the most important measure of success. The distribution of this reward is quite symmetrical, with a slightly higher peak in the positive area. The distribution of M2 shows that M1 and M2 are related. From the distribution of M4 we can conclude that the majority of dialogues during the training phase was not very engaging. Users obviously had a good feeling about what they could say at each point in the dialogue (M3), which implies good quality of the system prompts. The judgement about the usefulness of the provided advice is pretty average, tending a bit more to negative than to positive. We do think that this measure might be distorted by the fact that we asked the subjects to *imagine* that they have the given exercise barriers. Furthermore, they were sometimes confronted with advice that had already been presented to them in earlier conversations.

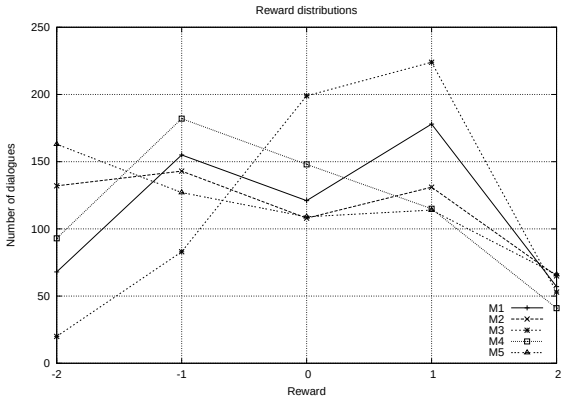


Figure 1: Reward distributions in the training data.

In our analysis of the users' rewarding behavior, we found several significant correlations. We found that longer dialogues (> 3 user turns) are appreciated more than short ones (< 4 user turns), which seems rather logical, as dialogues in which the user

barely gets to say anything are neither natural nor engaging.

We also looked at the relationship between user input verification and the given rewards. Our intuition is that the choice of barrier verification is one of the most important choices the system can make in the dialogue. We found that it is much better to first verify the detected barrier than to immediately give advice. The percentage of appropriate advice provided in dialogues with barrier verification is significantly higher than in dialogues without verification.

In several states of the dialogue, we let the system choose from different wordings of the system prompt. One of these choices is whether to use an open question to ask what the user's barrier is (*How can I help you?*), or a directive question (*Tell me what is preventing you from exercising more.*). The motivation behind the open question is that the user gets the initiative and is basically free to talk about anything he/she likes. Naturally, the advantage of directive questions is that the chance of making classification errors is much lower than with open questions because the user will be better able to assess what kind of answer the system expects. Dialogues in which the key-question (asking the user's barrier) was directive, were rewarded more positively than dialogues with the open question.

6.2 Learned dialogue policies

We learned a different policy for each evaluation measure separately (by only using the rewards given for that particular measure), and a policy based on a combination (sum) of the rewards for all evaluation measures. We found that the learned policy based on the combination of all measures, and the policy based on measure M1 alone (*Overall, this conversation went well*) were nearly identical. Table 4 compares the most important decisions of the different policies. For convenience of comparison, we only listed the main, structural choices. Table 3 shows which of the dialogue components in the library were used in the learned and the expert policy.

Note that, for the sake of clarity, the state descriptions in Table 4 are basically summaries of a set of more specific states since a state is a specific representation of the dialogue context at a particular moment (composed of the values of the features listed in Table 2). For instance, in the p_a policy, the decision in the last row of the table (give a joke or not), depends on whether or not there has been a classification failure (i.e. a communication problem earlier in the dialogue). If there has been a classification failure, the policy prescribes the decision *not* to give a joke, as it was not appreciated by the training users in that context. Otherwise, if there were no communication problems during the conversation, the users *did* appreciate a joke.

6.3 Evaluation

We compared the learned dialogue policy with a policy which was independently hand-designed by experts[2] for this system. The decisions made in the learned strategy were very similar to the ones made by the experts, with only a few differences, indicating that the automated method would indeed perform as well as an expert. The main differences were the inclusion of a personal questionnaire for relation building at the beginning of the dialogue and a commitment question at the end of the dialogue. Another difference was the more restricted use of the humour element, described in section 6.2 which turns out to be intuitively better than the expert's decision to simply always include a joke. Of course, we can only draw conclusions with regard to the effectiveness of these two policies if we empirically compare them with real test users. Such evaluations are planned as part of our future research.

As some additional evidence against the possibility that the learned policy was generated by chance, we performed a simple experiment in which we took several random samples of 300 training dialogues from the complete training set. For each sample, we learned the optimal policy. We mutually compared these policies and found that they were very similar: only in 15-20% of the states, the policies disagreed on which action to take next. On closer inspection we found that this disagreement mainly concerned states that were poorly visited (1-10 times) in these samples. These results suggest that the learned policy is unreliable at infrequently visited states. Note however, that all main decisions listed in Table 4 are

[2]The experts were a team made up of psychologists with experience in the psychology of health behaviour change and a scientist with experience in the design of automated dialogue systems.

State description	Action choices	p_1	p_2	p_3	p_4	p_5	p_a	p_e
After greeting the user	- ask the exercise barrier - ask personal information - chit-chat about exercise	•	•	•	•	•	•	•
When asking the barrier	- use a directive question - use an open question	•	•	•	•	•	•	•
User gives exercise barrier	- verify detected barrier - give solution	•	•	•	•	•	•	•
User rephrased barrier	- verify detected barrier - give solution	•	•	•	•	•	•	•
Before presenting solution	- ask if the user wants to see a solution for the barrier - give a solution	•	•	•	•	•	•	•
After presenting solution	- verify solution usefulness - encourage the user to work on the given solution - ask user to commit solution implementation	•	•	•	•	•	•	•
User found solution useful	- encourage the user to work on the solution - ask user to commit solution implementation	•	•	•	•	•	•	•
User found solution not useful	- give another solution - ask the user wants to propose his own solution	•	•	•	•	•	•	•
After giving second solution	- verify solution usefulness - encourage the user to work on the given solution - ask user to commit solution implementation	•	•	•	•	•	•	•
End of dialogue	- close the session - ask if the user wants to hear a joke	•	•	•	•	•	•	•

Table 4: Comparison of the most important decisions made by the learned policies. p_n is the policy based on evaluation measure n; p_a is the policy based on all measures; p_e contains the decisions made by experts in the manually designed policy.

made at frequently visited states. The only disagreement in frequently visited states concerned system-prompt choices. We might conclude that these particular (often very subtle) system-prompt choices (e.g. careful versus direct formulation of the exercise barrier) are harder to learn than the more noticable dialogue structure-related choices.

7 Conclusions and Future Work

We have explored reinforcement learning for automatic dialogue policy optimization in a question-based motivational dialogue system. Our system can automatically compose a dialogue strategy from a library of dialogue components, that is very similar to a manually designed expert strategy, by learning from user feedback.

Thus, in order to build a new dialogue system, dialogue system engineers will have to set up a rough dialogue template containing several 'multiple choice'-action nodes. At these nodes, various dialogue components or prompt wordings (e.g. entertaining parts, clarification questions, social dialogue, personal questions) from an existing or self-made library can be plugged in without knowing beforehand which of them would be most effective.

The automatically generated dialogue policy is very similar (see Table 4) –but arguably improved in many details– to the hand-designed policy for this system. Automatically learning dialogue policies also allows us to test a number of interesting issues in parallel, for example, we have learned that users appreciated dialogues that were longer, starting with some personal questions (e.g *What is your name?*, *What are your hobbies?*). We think that altogether, this relation building component gave the dialogue a more natural and engaging character, although it was left out in the expert strategy.

We think that the methodology described in this paper may be able to yield more effective dialogue policies than experts. Especially in complicated dialogue systems with large state spaces. In our system, state representations are composed of multiple context feature values (e.g. communication problem earlier in the dialogue, the confidence of the utterance classifier). Our experiments showed that sometimes different decisions were learned in dialogue contexts where only one of these features was different (for example use humour only if the system has been successful in recognising a user's exercise barrier): all context features are implicitly used to learn the optimal decisions and when hand-designing a di-

alogue policy, experts can impossibly take into account all possible different dialogue contexts.

With respect to future work, we plan to examine the impact of different state representations. We did not yet empirically compare the effects of each feature on policy learning or experiment with other features than the ones listed in Table 2. As Tetreault and Litman (2006) show, incorporating more or different information into the state representation might however result in different policies.

Furthermore, we will evaluate the actual genericity of our approach by applying it to different domains. As part of that, we will look at automatically mining libraries of dialogue components from existing dialogue transcript data (e.g. available scripts or transcripts of films, tv series and interviews containing real-life examples of different types of dialogue). These components can then be plugged into our current adaptive system in order to discover what works best in dialogue for new domains. We should note here that extending the system's dialogue component library will automatically increase the state space and thus policy generation and optimization will become more difficult and require more training data. It will therefore be very important to carefully control the size of the state space and the global structure of the dialogue.

Acknowledgements

The authors would like to thank Piroska Lendvai Rudenko, Walter Daelemans, and Bob Hurling for their contributions and helpful comments. We also thank the anonymous reviewers for their useful comments on the initial version of this paper.

References

Timothy W. Bickmore. 2003. *Relational Agents: Effecting Change through Human-Computer Relationships*. Ph.D. Thesis, MIT, Cambridge, MA.

Heriberto Cuayáhuitl, Steve Renals, Oliver Lemon, and Hiroshi Shimodaira. 2006. Learning multi-goal dialogue strategies using reinforcement learning with reduced state-action spaces. *Proceedings of Interspeech-ICSLP*.

Walter Daelemans, Sabine Buchholz, and Jorn Veenstra. 1999. Memory-Based Shallow Parsing. *Proceedings of CoNLL-99*, Bergen, Norway.

Michael S. English and Peter A. Heeman 2005. Learning Mixed Initiative Dialog Strategies By Using Reinforcement Learning On Both Conversants. *Proceedings of HLT/NAACL*.

Matthew Frampton and Oliver Lemon. 2006. Learning More Effective Dialogue Strategies Using Limited Dialogue Move Features. *Proceedings of the Annual Meeting of the ACL*.

James Henderson, Oliver Lemon, and Kallirroi Georgila. 2005. Hybrid Reinforcement/Supervised Learning for Dialogue Policies from COMMUNICATOR Data. *IJCAI workshop on Knowledge and Reasoning in Practical Dialogue Systems*.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 2000. A Stochastic Model of Human-Machine Interaction for Learning Dialog Strategies. *IEEE Trans. on Speech and Audio Processing*, Vol. 8, No. 1, pp. 11-23.

Diane J. Litman and Shimei Pan. 2002. Designing and Evaluating an Adaptive Spoken Dialogue System. *User Modeling and User-Adapted Interaction*, Volume 12, Issue 2-3, pp. 111-137.

Karen K. Liu and Rosalind W. Picard. 2005. Embedded Empathy in Continuous, Interactive Health Assessment. *CHI Workshop on HCI Challenges in Health Assessment*, Portland, Oregon.

Preetam Maloor and Joyce Chai. 2000. Dynamic User Level and Utility Measurement for Adaptive Dialog in a Help-Desk System. *Proceedings of the 1st Sigdial Workshop*.

Tim Paek and David M. Chickering. 2005. The Markov Assumption in Spoken Dialogue Management. *Proceedings of SIGDIAL 2005*.

Matthew Rudary, Satinder Singh, and Martha E. Pollack. 2004. Adaptive cognitive orthotics: Combining reinforcement learning and constraint-based temporal reasoning. *Proceedings of the 21st International Conference on Machine Learning*.

Konrad Scheffler and Steve Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. *Proceedings of HLT-2002*.

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System. *Journal of Artificial Intelligence Research (JAIR)*, Volume 16, pages 105-133.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning*. MIT Press.

Joel R. Tetreault and Diane J. Litman 2006. Comparing the Utility of State Features in Spoken Dialogue Using Reinforcement Learning. *Proceedings of HLT/NAACL*, New York.

Marilyn A. Walker 2000. An Application of Reinforcement Learning to Dialogue Strategy Selection in a Spoken Dialogue System for Email. *Journal of Artificial Intelligence Research*, Vol 12., pp. 387-416.

Jason D. Williams, Pascal Poupart, and Steve Young. 2005. Partially Observable Markov Decision Processes with Continuous Observations for Dialogue Management. *Proceedings of the 6th SigDial Workshop*, September 2005, Lisbon.

On the role of context and prosody in the interpretation of 'okay'

Agustín Gravano, Stefan Benus, Héctor Chávez, Julia Hirschberg, Lauren Wilcox
Department of Computer Science
Columbia University, New York, NY, USA
{agus,sbenus,hrc2009,julia,lgw23}@cs.columbia.edu

Abstract

We examine the effect of contextual and acoustic cues in the disambiguation of three discourse-pragmatic functions of the word *okay*. Results of a perception study show that contextual cues are stronger predictors of discourse function than acoustic cues. However, acoustic features capturing the pitch excursion at the right edge of *okay* feature prominently in disambiguation, whether other contextual cues are present or not.

1 Introduction

CUE PHRASES (also known as DISCOURSE MARKERS) are linguistic expressions that can be used to convey explicit information about the structure of a discourse **or** to convey a semantic contribution (Grosz and Sidner, 1986; Reichman, 1985; Cohen, 1984). For example, the word *okay* can be used to convey a 'satisfactory' evaluation of some entity in the discourse (*the movie was okay*); as a backchannel in a dialogue to indicate that one interlocutor is still attending to another; to convey acknowledgment or agreement; or, in its 'cue' use, to start or finish a discourse segment (Jefferson, 1972; Schegloff and Sacks, 1973; Kowtko, 1997; Ward and Tsukahara, 2000). A major question is how speakers indicate and listeners interpret such variation in meaning. From a practical perspective, understanding how speakers and listeners disambiguate cue phrases is important to spoken dialogue systems, so that systems can convey potentially ambiguous terms with their intended meaning and can interpret user input correctly.

There is considerable evidence that the different uses of individual cue phrases can be distinguished by variation in the prosody with which they are realized. For example, (Hirschberg and Litman, 1993) found that cue phrases in general could be disambiguated between their 'semantic' and their 'discourse marker' uses in terms of the type of pitch accent borne by the cue phrase, the position of the phrase in the intonational phrase, and the amount of additional information in the phrase. Despite the frequence of the word *okay* in natural dialogues, relatively little attention has been paid to the relationship between its use and its prosodic realization. (Hockey, 1993) did find that *okay* differs in terms of the pitch contour speakers use in uttering it, suggesting that a final rising pitch contour "categorically marks a turn change," while a downstepped falling pitch contour usually indicates a discourse segment boundary. However, it is not clear which, if any, of the prosodic differences identified in this study are actually used by listeners in interpreting these potentially ambiguous items.

In this study, we address the question of how hearers disambiguate the interpretation of *okay*. Our goal is to identify the acoustic, prosodic and phonetic features of *okay* tokens for which listeners assign different meanings. Additionally, we want to determine the role that discourse context plays in this classification: i.e., can subjects classify *okay* tokens reliably from the word alone or do they require additional context?

Below we describe a perception study in which listeners were presented with a number of spoken productions of *okay*, taken from a corpus of dialogues between subjects playing a computer game. The tokens were presented both in isolation and in context. Users were asked to select the meaning

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 800–807,
Prague, Czech Republic, June 2007.

of each token from three of the meanings that *okay* can take on: ACKNOWLEDGEMENT/AGREEMENT, BACKCHANNEL, and CUE OF AN INITIAL DISCOURSE SEGMENT. Subsequently, we examined the acoustic, prosodic and phonetic correlates of these classifications to try to infer what cues listeners used to interpret the tokens, and how these varied by context condition. Section 2 describes our corpus. Section 3 describes the perception experiment. In Section 4 we analyze inter-subject agreement, introduce a novel representation of subject judgments, and examine the acoustic, prosodic, phonetic and contextual correlates of subject classification of *okay*s. In Section 5 we discuss our results and future work.

2 Corpus

The materials for our perception study were selected from a portion of the Columbia Games Corpus, a collection of 12 spontaneous task-oriented dyadic conversations elicited from speakers of Standard American English. The corpus was collected and annotated jointly by the Spoken Language Group at Columbia University and the Department of Linguistics at Northwestern University.

Subjects were paid to play two series of computer games (the CARDS GAMES and the OBJECTS GAMES), requiring collaboration between partners to achieve a common goal. Participants sat in front of laptops in a soundproof booth with a curtain between them, so that all communication would be verbal. Each player played with two different partners in two different sessions. On average, each session took 45m 39s, totalling 9h 8m of dialogue for the whole corpus. All interactions were recorded, digitized, and downsampled to 16K.

The recordings were orthographically transcribed and words were aligned by hand by trained annotators in a ToBI (Beckman and Hirschberg, 1994) orthographic tier using Praat (Boersma and Weenink, 2001) to manipulate waveforms. The corpus contains 2239 unique words, with 73,831 words in total. Nearly all of the Objects Games part of the corpus has been intonationally transcribed, using the ToBI conventions. Pitch, energy and duration information has been extracted for the entire corpus automatically, using Praat.

In the Objects Games portion of the corpus each player's laptop displayed a gameboard containing 5–7 objects (Figure 1). In each segment of the game, both players saw the same set of objects at the same position on each screen, except for one object (the TARGET). For one player (the DESCRIBER), this target appeared in a random location among other objects on the screen. For the other player (the FOLLOWER), the target object appeared at the bottom of the screen. The describer was instructed to describe the position of the target object on their screen so that the follower could move their representation of the target to the same location on their own screen. After the players had negotiated what they determined to be the best location, they were awarded up to 100 points based on the actual match of the target location on the two screens. The game proceeded in this way through 14 tasks, with describer and follower alternating roles. On average, the Objects Games portion of each session took 21m 36s, resulting in 4h 19m of dialogue for the twelve sessions in the corpus. There are 1484 unique words in this portion of the corpus, and 36,503 words in total.

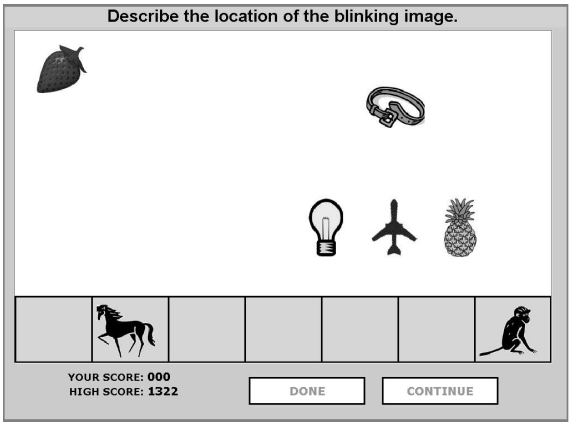


Figure 1: Sample screen of the Objects Games.

Throughout the Objects Games, we noted that subjects made frequent use of affirmative cue words, such as *okay*, *yeah*, *alright*, which appeared to vary in meaning. To investigate the discourse functions of such words, we first asked three labelers to independently classify all occurrences of *alright, gotcha, huh, mmhm, okay, right, uhhuh, yeah, yep, yes, yup* in the entire Games Corpus into one of ten categories, including acknowledgment/agreement, cue beginning or ending discourse segment, backchannel, and literal modifier. Labelers were asked to

choose the most appropriate category for each token, or indicate with '?' if they could not make a decision. They were allowed to read the transcripts and listen to the speech as they labeled.

For our perception experiment we chose materials from the tokens of the most frequent of our labeled affirmative words, *okay*, from the Objects Games, which contained most of these tokens. Altogether, there are 1151 instances of *okay* in this part of the corpus; it is the third most frequent word, following *the*, with 4565 instances, and *of*, with 1534. At least two labelers agreed on the functional category of 902 (78%) *okay* tokens. Of those tokens, 286 (32%) were classified as BACKCHANNEL, 255 (28%) as ACKNOWLEDGEMENT/AGREEMENT, 141 (16%) as CUE BEGINNING, 116 (13%) as PIVOT BEGINNING (a function that combines Acknowledgement/agreement and Cue beginning), and 104 (11%) as one of the other functions. We sampled from tokens the annotators had labeled as Cue beginning discourse segment, Backchannel, and Acknowledgement/agreement, the most frequent categories in the corpus; we will refer to these below simply as 'C', 'B', and 'A' classes, respectively.

3 Experiment

We next designed a perception experiment to examine naive subjects' perception of these tokens of *okay*. To obtain good coverage both of the (labeled) A, B, and C classes, as well as the degrees of potential ambiguity among these classes, we identified 9 categories of *okay* tokens to include in the experiment: 3 classes (A, B, C) $\times$ 3 levels of labeler agreement (UNANIMOUS, MAJORITY, NO-AGREEMENT). 'Unanimous' refers to tokens assigned to a particular class label by all 3 labelers, 'majority' to tokens assigned to this class by 2 of the 3 labelers, and 'no-agreement' to tokens assigned to this class by only 1 labeler. To decrease variability in the stimuli, we selected tokens only from speakers who produced at least one token for each of the 9 conditions. There were 6 such speakers (3 female, 3 male), which gave us a total of 54 tokens.

To see whether subjects' classifications of *okay* were dependent upon contextual information or not, we prepared two versions of each token. The isolated versions consisted of only the word *okay* extracted from the waveform. For the contextualized versions, we extracted two full speaker turns for each *okay* including the full turn[1] containing the target *okay* plus the full turn of the previous speaker. In the following three sample contexts, pauses are indicated with '#', and the target *okay*s are underlined:

Speaker A: yeah # um there's like there's some space there's
Speaker B: okay # I think I got it

Speaker A: but it's gonna be below the onion
Speaker B: okay

Speaker A: okay # alright # I'll try it # okay
Speaker B: okay the owl is blinking

The isolated *okay* tokens were single channel audio files; the contextualized *okay* tokens were formatted so that each speaker was presented to subjects on a different channel, with the speaker uttering the target *okay* consistently on the same channel.

The perception study was divided into two parts. In the first part, each subject was presented with the 54 isolated *okay* tokens, in a different random ordering for each subject. They were given a forced choice task to classify them as A, B, or C, with the corresponding labels (Acknowledgement/agreement, Backchannel, and Cue beginning) also presented in a random order for each token. In the second part, the same subject was given 54 contextualized tokens, presented in a different random order, and asked to make the same choice.

We recruited 20 (paid) subjects for the study, 10 female, and 10 male, all between the ages of 20 and 60. All subjects were native speakers of Standard American English, except for one subject who was born in Jamaica but a native speaker of English. All subjects reported no hearing problems. Subjects performed the study in a quiet lab using headphones to listen to the tokens and indicating their classification decisions in a GUI interface on a lab workstation. They were given instructions on how to use the interface before each of the two sections of the study.

For the study itself, for each token in the **isolated** condition, subjects were shown a screen with the three randomly ordered classes and a link to the token's sound file. They could listen to the sound files as many times as they wished but were instructed not to be concerned with answering the questions

[1] We define a TURN as a maximal sequence of words spoken by the same speaker during which the speaker holds the floor.

"correctly", but to answer with their immediate response if possible. However, they were allowed to change their selection as many times as they liked before moving to the next screen. In the **contextualized** condition, they were also shown an orthographic transcription of part of the contextualized token, to help them identify the target *okay*. The mean duration of the first part of the study was 25 minutes, and of the second part, 27 minutes.

4 Results

4.1 Subject ratings

The distribution of class labels in each experimental condition is shown in Table 1. While this distribution roughly mirrors our selection of equal numbers of tokens from each previously-labeled class, in both parts of the study more tokens were labeled as A (*acknowledgment/agreement*) than as B (*backchannel*) or C (*cue to topic beginning*). This supports the hypothesis that *acknowledgment/agreement* may function as the default interpretation of *okay*.

	Isolated	**Contextualized**
A	426 (39%)	452 (42%)
B	324 (30%)	306 (28%)
C	330 (31%)	322 (30%)
Total	1080 (100%)	1080 (100%)

Table 1: Distribution of label classes in each study condition.

We examined inter-subject agreement using Fleiss' κ measure of inter-rater agreement for multiple raters (Fleiss, 1971).[2] Table 2 shows Fleiss' κ calculated for each individual label vs. the other two labels and for all three labels, in both study conditions. From this table we see that, while there is very little overall agreement among subjects about how to classify tokens in the **isolated** condition, agreement is higher in the **contextualized** condition, with a moderate agreement for class C (κ score of .497). This suggests that context helps distinguish the *cue beginning discourse segment* function more than the other two functions of *okay*.

[2] This measure of agreement above chance is interpreted as follows: 0 = None, 0 - 0.2 = Small, 0.2 - 0.4 = Fair, 0.4 - 0.6 = Moderate, 0.6 - 0.8 = Substantial, 0.8 - 1 = Almost perfect.

	Isolated	**Contextualized**
A vs. rest	.089	.227
B vs. rest	.118	.164
C vs. rest	.157	.497
all	.120	.293

Table 2: Fleiss' κ for each label class in each study condition.

Recall from Section 3 that the *okay* tokens were chosen in equal numbers from three classes according to the level of agreement of our three original labelers (unanimous, majority, and no-agreement), who had the full dialogue context to use in making their decisions. Table 3 shows Fleiss' κ measure now grouped by amount of agreement of the original labelers, again presented for each context condition. We see here that the inter-subject agreement

	Isolated	**Context.**	**OL**
no-agreement	.085	.104	-
majority	.092	.299	-
unanimous	.158	.452	-
all	.120	.293	.312

Table 3: Fleiss' κ in each study condition, grouped by agreement of the three original labelers ('OL').

also mirrors the agreement of the three original labelers. In both study conditions, tokens which the original labelers agreed on also had the highest κ scores, followed by tokens in the majority and no-agreement classes, in that order. In all cases, tokens which subjects heard in context showed more agreement than those they heard in isolation.

The overall κ is small at .120 for the **isolated** condition, and fair at .293 for the **contextualized** condition. The three original labelers also achieved fair agreement at .312.[3] The similarity between the latter two κ scores suggests that the full context available to the original labelers and the limited context presented to the experiment subjets offer comparable amounts of information to disambiguate between the three functions, although lack of any context clearly affected subjects' decisions. We conclude

[3] For the calculation of this κ, we considered four label classes: A, B, C, and a fourth class 'other' that comprises the remaining 7 word functions mentioned in Section 2. In consequence, these κ scores should be compared with caution.

from these results that context is of considerable importance in the interpretation of the word *okay*, although even a very limited context appears to suffice.

4.2 Representing subject judgments

In this section, we present a graphical representation of subject decisions, useful for interpreting, visualizing, and comparing the way our subjects interpreted the different tokens of *okay*. For each individual *okay* in the study, we define an associated three-dimensional VOTE VECTOR, whose components are the proportions of subjects that classified the token as A, B or C. For example, if a particular *okay* was labeled as A by 5 subjects, as B by 3, and as C by 12, then its associated vote vector is $\left(\frac{5}{20}, \frac{3}{20}, \frac{12}{20}\right) = (0.25, 0.15, 0.6)$. Following this definition, the vectors $\mathcal{A} = (1, 0, 0)$, $\mathcal{B} = (0, 1, 0)$ and $\mathcal{C} = (0, 0, 1)$ correspond to the ideal situations in which all 20 subjects agreed on the label. We call these vectors the UNANIMOUS-VOTE VECTORS.

Figure 2.i shows a two-dimensional representation that illustrates these definitions. The black dot

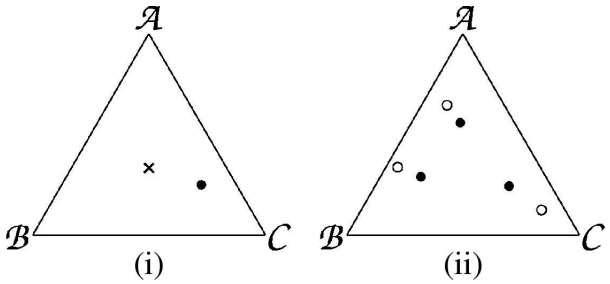


Figure 2: 2D representation of a vote vector (i) and of the cluster centroids (ii).

represents the vote vector for our example *okay*, the vertices of the triangle correspond to the three unanimous-vote vectors ($\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$), and the cross in the center of the triangle represents the vote vector of a three-way tie between the labelers $\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$.

We are thus able to calculate the Euclidean distance of a vote vector to each of the unanimous-vote vectors. The shortest of these distances corresponds to the label assigned by the plurality[4] of subjects. Also, the smaller that distance, the higher the inter-subject agreement for that particular token. For our example *okay*, the distances to $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ are 0.972, 1.070 and 0.495, respectively; its plurality label is C.

In our experiment, each *okay* has two associated vote vectors, one for each context condition. To illustrate the relationship between decisions in the **isolated** and the **contextualized** conditions, we first grouped each condition's 54 vote vectors into three clusters, according to their plurality label. Figure 2.ii shows the cluster centroids in a two-dimensional representation of vote vectors. The filled dots correspond to the cluster centroids of the **isolated** condition, and the empty dots, to the centroids of the **contextualized** condition. Table 4 shows the distances in each condition from the cluster centroids (denoted A_c, B_c, C_c) to the respective unanimous-vote vectors ($\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$), and also the distance between each pair of cluster centroids.

	Isolated	**Contextualized**
$d(A_c, \mathcal{A})$	.54	.44 (–18%)
$d(B_c, \mathcal{B})$	.57	.52 (–10%)
$d(C_c, \mathcal{C})$	.52	.28 (–47%)
$d(A_c, B_c)$	.41	.48 (+17%)
$d(A_c, C_c)$	.49	.86 (+75%)
$d(B_c, C_c)$	.54	.91 (+69%)

Table 4: Distances from the cluster centroids (A_c, B_c, C_c) to the unanimous-vote vectors ($\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$) and between cluster centroids, in each condition.

In the **isolated** condition, the three cluster centroids are approximately equidistant from each other —that is, the three word functions appear to be equally confusable. In the **contextualized** condition, while C_c is further apart from the other two centroids, the distance between A_c and B_c remains practically the same. This suggests that, with some context available, A and B tokens are still fairly confusable, while both are more easily distinguished from C tokens. We posit two possible explanations for this: First, C is the only function for which the speaker uttering the *okay* necessarily continues speaking; thus the role of context in disambiguating seems quite clear. Second, both A and B have a common element of 'acknowledgement' that might affect inter-subject agreement.

[4] *Plurality* is also known as *simple majority*: the candidate who gets more votes than any other candidate is the winner.

4.3 Features of the *okay* tokens

In this section, we describe a set of acoustic, prosodic, phonetic and contextual features which may help to explain why subjects interpret *okay* differently. Acoustic features were extracted automatically using Praat. Phonetic and prosodic features were hand-labeled by expert annotators. Contextual features were considered only in the analysis of the **contextualized** condition, since they were not available to subjects in the **isolated** condition.

We examined a number of phonetic features to determine whether these correlated with subject classifications. We first looked at the production of the three phonemes in the target *okay* (/oʊ/, /k/, /eɪ/), noting the following possible variations:

- /oʊ/: [], [ɑ], [ɐ], [ɔ], [ɔʊ], [m], [ŋ], [ə], [əʊ].
- /k/: [ɣ], [k], [kx], [q], [x].
- /eɪ/: [e], [eɪ], [ɛ], [eə].

We also calculated the duration of each phone and of the velar closure. Whether the target *okay* was at least partially whispered or not, and whether there was glottalization in the target *okay* were also noted.

For each target *okay*, we also examined its duration and its maximum, mean and minimum pitch and intensity, as well as the speaker-normalized versions of these values.[5] We considered its pitch slope, intensity slope, and stylized pitch slope, calculated over the whole target *okay*, its last 50, 80 and 100 milliseconds, its second half, its second syllable, and the second half of its second syllable, as well.

We used the ToBI labeling scheme (Pitrelli et al., 1994) to label the prosody of the target *okay*s and their surrounding context.

- Pitch accent, if any, of the target *okay* (e.g., H*, H+!H*, L*).
- Break index after the target *okay* (0-4).
- Phrase accent and boundary tone, if any, following the target *okay* (e.g., L-L%, !H-H%).

For contextualized tokens, we included several features related to the exchange between the speaker uttering the target *okay* (*Speaker B*) and the other speaker (*Speaker A*).

- Number of words uttered by *Speaker A* in the context, before and after the target *okay*. Same for *Speaker B*.
- Latency of *Speaker A* before *Speaker B*'s turn.
- Duration of silence of *Speaker B* before and after the target *okay*.
- Duration of speech by *Speaker B* immediately before and after the target *okay* and up to a silence.

4.4 Cues to interpretation

We conducted a series of Pearson's tests to look for correlations between the proportion of subjects that chose each label and the numeric features described in Section 4.3, together with two-sided t-tests to find whether such correlations differed significantly from zero. Tables 5 and 6 show the significant results (two-sided t-tests, $p < 0.05$) for the **isolated** and **contextualized** conditions, respectively.

Acknowledgement/agreement	r
duration of realization of /k/	–0.299
Backchannel	r
stylized pitch slope over 2nd half 2nd syl.	0.752
pitch slope over 2nd half of 2nd syllable	0.409
speaker-normalized maximum intensity	–0.372
pitch slope over last 80 ms	0.349
speaker-normalized mean intensity	–0.327
duration of realization of /eɪ/	0.278
word duration	0.277
Cue to discourse segment beginning	r
stylized pitch slope over the whole word	–0.380
pitch slope over the whole word	–0.342
pitch slope over 2nd half of 2nd syllable	–0.319

Table 5: Features correlated to the proportion of votes for each label. **Isolated** condition.

Table 5 shows that in the **isolated** condition, subjects tended to classify tokens of *okay* as Acknowledgment/agreement (A) which had a longer realization of the /k/ phoneme. They tended to classify tokens as Backchannels (B) which had a lower intensity, a longer duration, a longer realization of the /eɪ/ phoneme, and a final rising pitch. They tended to classify tokens as C (cue to topic beginning) that ended with falling pitch.

[5] Speaker-normalized features were normalized by computing z-scores ($z = (X - mean)/st.dev$) for the feature, where $mean$ and $st.dev$ were calculated from all *okay*s uttered by the speaker in the session.

Acknowledgement/agreement	r
latency of *Spkr A* before *Spkr B*'s turn	–0.528
duration of silence by *Spkr B* before *okay*	–0.404
number of words by *Spkr B* after *okay*	–0.277
Backchannel	r
pitch slope over 2nd half of 2nd syllable	0.520
pitch slope over last 80 ms	0.455
number of words by *Spkr A* before *okay*	0.451
number of words by *Spkr B* after *okay*	–0.433
duration of speech by *Spkr B* after *okay*	–0.413
latency of *Spkr A* before *Spkr B*'s turn	–0.385
duration of silence by *Spkr B* before *okay*	0.295
intensity slope over 2nd syllable	–0.279
Cue to discourse segment beginning	r
latency of *Spkr A* before *Spkr B*'s turn	0.645
number of words by *Spkr B* after *okay*	0.481
number of words by *Spkr A* before *okay*	–0.426
pitch slope over 2nd half of 2nd syllable	–0.385
pitch slope over last 80 ms	–0.377
duration of speech by *Spkr B* after *okay*	0.338

Table 6: Features correlated to the proportion of votes for each label. **Contextualized** condition.

In the **contextualized** condition, we find very different correlations. Table 6 shows that nearly all of the strong correlations in this condition involve contextual features, such as the latency before *Speaker B*'s turn, or the number of words by each speaker before and after the target *okay*. Notably, only one of the features that show strong correlations in the **isolated** condition shows the same strong correlation in the **contextualized** condition: the pitch slope at the end of the word. In both conditions subjects tended to label tokens with a final rising pitch contour as B, and tokens with a final falling pitch contour as C. This supports (Hockey, 1993)'s findings on the role of pitch contour in disambiguating *okay*.

We next conducted a series of two-sided Fisher's exact tests to find correlations between subjects' labelings of *okay* and the nominal features described in Section 4.3. We found significant associations between the realization of the /oʊ/ phoneme and the *okay* function in the **isolated** condition ($p < 0.005$). Table 7 shows that, in particular, [m] seems to be the preferred realization for B *okay*s, while [ə] seems to be the preferred one for A *okay*s, and [ɔʊ] and [ɔ] for A and C *okay*s.

	?	[ɑ]	[ɐ]	[ɔʊ]	[ɔ]	[ŋ]	[əʊ]	[ə]	[]	[m]
A	0	0	5	6	4	0	0	8	0	0
B	2	0	4	1	0	1	0	1	1	5
C	1	1	2	3	4	0	1	3	0	0

Table 7: Realization of the /oʊ/ phoneme, grouped by subject plurality label. **Isolated** condition only.

Notably, we did not find such significant associations in the **contextualized** condition. We did find significant correlations in both conditions, however, between *okay* classifications and the type of phrase accent and boundary tone following the target (Fisher's Exact Test, $p < 0.05$ for the **isolated** condition, $p < 0.005$ for the **contextualized** condition). Table 8 shows that L-L% tends to be associated with A and C classes, H-H% with B classes, and L-H% with A and B classes. In this case, such correlations are present in the **isolated** condition, and sustained or enhanced in the **contextualized** condition.

		H-H%	H-L%	L-H%	L-L%	other
Isolated	A	0	2	4	8	9
	B	3	3	1	5	3
	C	1	1	0	8	5
Context.	A	0	2	3	10	10
	B	4	3	2	1	2
	C	0	1	0	10	5

Table 8: Phrase accent and boundary tone, grouped by subject plurality label.

Summing up, when subjects listened to the *okay* tokens in isolation, with only their acoustic, prosodic and phonetic properties available, a few features seem to strongly correlate with the perception of word function; for example, maximum intensity, word duration, and realizing the /oʊ/ phoneme as [m] tend to be associated with *backchannel*, while the duration of the realization of the /k/ phoneme, and realizing the /oʊ/ phoneme as [ə] tend to be associated with *acknowledgment/agreement*.

In the second part of the study, when subjects listened to contextualized versions of the same tokens of *okay*, most of the strong correlations of word function with acoustic, prosodic and phonetic features were replaced by correlations with contextual features, like latency and turn duration. In other words, these results suggest that contextual features

might override the effect of most acoustic, prosodic and phonetic features of *okay*. There is nonetheless one notable exception: word final intonation — captured by the pitch slope and the ToBI labels for phrase accent and boundary tone — seems to play a central role in the interpretation of both isolated and contextualized *okay*s.

5 Conclusion and future work

In this study, we have presented evidence of differences in the interpretation of the function of isolated and contextualized *okay*s. We have shown that word final intonation strongly correlates with the subjects' classification of *okay*s in both conditions. Additionally, the higher degree of inter-subject agreement in the contextualized condition, along with the strong correlations found for contextualized features, suggests that context, when available, plays a central role in the disambiguation of *okay*. (Note, however, that further research is needed in order to assess whether these features are indeed, in fact, perceptually important, both individually and combined.)

We have also presented results suggesting that *acknowledgment/agreement* acts as a default function for both isolated an contextualized *okay*s. Furthermore, while that function remains confusable with *backchannel* in both conditions, the availability of some context helps in distinguishing those two functions from *cue to topic beginning*.

These results are relevant to spoken dialogue systems in suggesting how systems can convey the cue word *okay* with the intended meaning and can interpret users' productions of *okay* correctly. How these results extend to other cue words and to other word functions remains an open question.

As future work, we will extend this study to include the over 5800 occurrences of *alright, gotcha, huh, mmhm, okay, right, uhhuh, yeah, yep, yes, yup* in the entire Games Corpus, and all 10 discourse functions mentioned in Section 2, as annotated by our three original labelers. Since we have observed considerable differences in conversation style in the two parts of the corpus (the Objects Games elicited more 'dynamic' conversations, with more overlaps and interruptions than the Cards Games), we will compare cue phrase usage in these two settings. Finally, we are also interested in examining speaker entrainment in cue phrase usage, or how subjects adapt their choice and production of cue phrases to their conversation partner's.

Acknowledgments

This work was funded in part by NSF IIS-0307905. We thank Gregory Ward, Elisa Sneed, and Michael Mulley for their valuable help in collecting and labeling the data, and the anonymous reviewers for helpful comments and suggestions.

References

Mary E. Beckman and Julia Hirschberg. 1994. The ToBI annotation conventions. *Ohio State University*.

Paul Boersma and David Weenink. 2001. Praat: Doing phonetics by computer. *http://www.praat.org*.

Robin Cohen. 1984. A computational theory of the function of clue words in argument understanding. *22nd Conference of the ACL*, pages 251–258.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.

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

Julia Hirschberg and Diane Litman. 1993. Empirical Studies on the Disambiguation of Cue Phrases. *Computational Linguistics*, 19(3):501–530.

Beth Ann Hockey. 1993. Prosody and the role of okay and uh-huh in discourse. *Proceedings of the Eastern States Conference on Linguistics*, pages 128–136.

Gail Jefferson. 1972. Side sequences. *Studies in social interaction*, 294:338.

Jacqueline C. Kowtko. 1997. *The function of intonation in task-oriented dialogue*. Ph.D. thesis, University of Edinburgh.

John Pitrelli, Mary Beckman, and Julia Hirschberg. 1994. Evaluation of prosodic transcription labeling reliability in the ToBI framework. In *ICSLP94*, volume 2, pages 123–126, Yokohama, Japan.

Rachel Reichman. 1985. *Getting Computers to Talk Like You and Me: Discourse Context, Focus, and Semantics: (an ATN Model)*. MIT Press.

Emanuel A. Schegloff and Harvey Sacks. 1973. Opening up closings. *Semiotica*, 8(4):289–327.

Nigel Ward and Wataru Tsukahara. 2000. Prosodic features which cue back-channel responses in English and Japanese. *Journal of Pragmatics*, 23:1177–1207.

Predicting Success in Dialogue

David Reitter and Johanna D. Moore
dreitter | jmoore @ inf.ed.ac.uk
School of Informatics
University of Edinburgh
United Kingdom

Abstract

Task-solving in dialogue depends on the linguistic alignment of the interlocutors, which Pickering & Garrod (2004) have suggested to be based on mechanistic repetition effects. In this paper, we seek confirmation of this hypothesis by looking at repetition in corpora, and whether repetition is correlated with task success. We show that the relevant repetition tendency is based on slow adaptation rather than short-term priming and demonstrate that lexical and syntactic repetition is a reliable predictor of task success given the first five minutes of a task-oriented dialogue.

1 Introduction

While humans are remarkably efficient, flexible and reliable communicators, we are far from perfect. Our dialogues differ in how successfully information is conveyed. In task-oriented dialogue, where the interlocutors are communicating to solve a problem, task success is a crucial indicator of the success of the communication.

An automatic measure of task success would be useful for evaluating conversations among humans, e.g., for evaluating agents in a call center. In human-computer dialogues, predicting the task success after just a first few turns of the conversation could avoid disappointment: if the conversation isn't going well, a caller may be passed on to a human operator, or the system may switch dialogue strategies. As a first step, we focus on human-human dialogue, since current spoken dialogue systems do not yet yield long, syntactically complex conversations.

In this paper, we use syntactic and lexical features to predict task success in an environment where we assume no speaker model, no semantic information and no information typical for a human-computer dialogue system, e.g., ASR confidence. The features we use are based on a psychological theory, linking alignment between dialogue participants to low-level syntactic priming. An examination of this priming reveals differences between short-term and long-term effects.

1.1 Repetition supports dialogue

In their *Interactive Alignment Model* (IAM), Pickering and Garrod (2004) suggest that dialogue between humans is greatly aided by *aligning* representations on several linguistic and conceptual levels. This effect is assumed to be driven by a cascade of linguistic priming effects, where interlocutors tend to re-use lexical, syntactic and other linguistic structures after their introduction. Such reuse leads speakers to agree on a common situation model. Several studies have shown that speakers copy their interlocutor's syntax (Branigan et al., 1999). This effect is usually referred to as *structural* (or: *syntactic*) *priming*. These persistence effects are inter-related, as lexical repetition implies preferences for syntactic choices, and syntactic choices lead to preferred semantic interpretations. Without demanding additional cognitive resources, the effects form a causal chain that will benefit the interlocutor's purposes. Or, at the very least, it will be easier for them to repeat linguistic choices than to

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 808–815,
Prague, Czech Republic, June 2007.

actively discuss their terminology and keep track of each other's current knowledge of the situation in order to come to a mutual understanding.

1.2 Structural priming

The repetition effect at the center of this paper, priming, is defined as a tendency to repeat linguistic decisions. Priming has been shown to affect language production and, to a lesser extent, comprehension, at different levels of linguistic analysis. This tendency may show up in various ways, for instance in the case of lexical priming as a shorter response time in lexical decision making tasks, or as a preference for one syntactic construction over an alternative one in syntactic priming (Bock, 1986). In an experimental study (Branigan et al., 1999), subjects were primed by completing either sentence (1a) or (1b):

1a. *The racing driver showed the torn overall...*
1b. *The racing driver showed the helpful mechanic...*

Sentence (1a) was to be completed with a prepositional object ("to the helpful mechanic"), while (1b) required a double object construction ("the torn overall"). Subsequently, subjects were allowed to freely complete a sentence such as the following one, describing a picture they were shown:

2. *The patient showed ...*

Subjects were more likely to complete (2) with a double-object construction when primed with (1b), and with a prepositional object construction when primed with (1a).

In a previous corpus-study, using transcriptions of spontaneous, task-oriented and non-task-oriented dialogue, utterances were annotated with syntactic trees, which we then used to determine the phrase-structure rules that licensed production (and comprehension) of the utterances (Reitter et al., 2006b). For each rule, the time of its occurrence was noted, e.g. we noted

3. 117.9*s NP* $\rightarrow$ *AT AP NN a fenced meadow*
4. 125.5*s NP* $\rightarrow$ *AT AP NN the abandoned cottage*

In this study, we then found that the re-occurrence of a rule (as in 4) was correlated with the temporal distance to the first occurrence (3), e.g., 7.6 seconds. The shorter the distance between prime (3) and target (4), the more likely were rules to re-occur.

In a conversation, priming may lead a speaker to choose a verb over a synonym because their interlocutor has used it a few seconds before. This, in turn, will increase the likelihood of the structural form of the arguments in the governed verbal phrase–simply because lexical items have their preferences for particular syntactic structures, but also because structural priming may be stronger if lexical items are repeated (lexical boost, Pickering and Branigan (1998)). Additionally, the structural priming effects introduced above will make a previously observed or produced syntactic structure more likely to be re-used. This chain reaction leads interlocutors in dialogue to reach a common situation model. Note that the IAM, in which interlocutors automatically and cheaply build a common representation of common knowledge, is at odds with views that afford each dialogue participant an explicit and separate representation of their interlocutor's knowledge.

The connection between linguistic persistence or priming effects and the success of dialogue is crucial for the IAM. The predictions arising from this, however, have eluded testing so far. In our previous study (Reitter et al., 2006b), we found more syntactic priming in the task-oriented dialogues of the Map Task corpus than in the spontaneous conversation collected in the Switchboard corpus. However, we compared priming effects across two datasets, where participants and conversation topics differed greatly. Switchboard contains spontaneous conversation over the telephone, while the task-oriented Map Task corpus was recorded with interlocutors co-present. While the result (more priming in task-oriented dialogue) supported the predictions of IAM, cognitive load effects could not be distinguished from priming. In the current study, we examine structural repetition in task-oriented dialogue only and focus on an extrinsic measure, namely task success.

2 Related Work

Prior work on predicting task success has been done in the context of human-computer spoken dialogue systems. Features such as recognition error rates, natural language understanding confidence and context shifts, confirmations and re-prompts (dialogue management) have been used classify dia-

logues into *successful* and *problematic* ones (Walker et al., 2000). With these automatically obtainable features, an accuracy of 79% can be achieved given the first two turns of "How may I help you?" dialogues, where callers are supposed to be routed given a short statement from them about what they would like to do. From the whole interaction (very rarely more than five turns), 87% accuracy can be achieved (36% of dialogues had been hand-labeled "problematic"). However, the most predictive features, which related to automatic speech recognition errors, are neither available in the human-human dialogue we are concerned with, nor are they likely to be the cause of communication problems there.

Moreover, failures in the Map Task dialogues are due to the actual goings-on when two interlocutors engage in collaborative problem-solving to jointly reach an understanding. In such dialogues, interlocutors work over a period of about half an hour. To predict their degree of success, we will leverage the phenomenon of *persistence*, or *priming*.

In previous work, two paradigms have seen extensive use to measure repetition and priming effects. *Experimental studies* expose subjects to a particular syntactic construction, either by having them produce the construction by completing a sample sentence, or by having an experimenter or confederate interlocutor use the construction. Then, subjects are asked to describe a picture or continue with a given task, eliciting the target construction or a competing, semantically equivalent alternative. The analysis then shows an effect of the controlled condition on the subject's use of the target construction.

Observational studies use naturalistic data, such as text and dialogue found in corpora. Here, the prime construction is not controlled–but again, a correlation between primes and targets is sought. Specific competing constructions such as active/passive, verbal particle placement or *that*-deletion in English are often the object of study (Szmrecsanyi, 2005; Gries, 2005; Dubey et al., 2005; Jäger, 2006), but the effect can also be generalized to syntactic phrase-structure rules or combinatorial categories (Reitter et al., 2006a).

Church (2000) proposes adaptive language models to account for lexical adaptation. Each document is split into *prime* and *target* halves. Then, for selected words w, the model estimates

$$P(+adapt) = P(w \in target | w \in prime)$$

$P(+adapt)$ is higher than $P_{prior} = P(w \in target)$, which is not surprising, since texts are usually about a limited number of topics.

This method looks at repetition over whole document halves, independently of decay. In this paper, we apply the same technique to syntactic rules, where we expect to estimate syntactic priming effects of the long-term variety.

3 Repetition-based Success Prediction

3.1 The Success Prediction Task

In the following, we define two variants of the task and then describe a model that uses repetition effects to predict success.

Task 1: *Success is estimated* when an entire dialogue is given. All linguistic and non-linguistic information available may be used. This task reflects post-hoc analysis applications, where dialogues need to be evaluated without the actual success measure being available for each dialogue. This covers cases where, e.g., it is unclear whether a call center agent or an automated system actually responded to the call satisfactorily.

Task 2: *Success is predicted* when just the initial 5-minute portion of the dialogue is available. A dialogue system's or a call center agent's strategy may be influenced depending on such a prediction.

3.2 Method

To address the tasks described in the previous Section, we train support vector machines (SVM) to predict the task success score of a dialogue from lexical and syntactic repetition information accumulated up to a specified point in time in the dialogue.

Data

The HCRC Map Task corpus (Anderson et al., 1991) contains 128 dialogues between subjects, who were given two slightly different maps depicting the same (imaginary) landscape. One subject gives directions for a predefined route to another subject, who follows them and draws a route on their map.

The spoken interactions were recorded, transcribed and syntactically annotated with phrase-structure grammar.

The Map Task provides us with a precise measure of success, namely the deviation of the predefined and followed route. Success can be quantified by computing the inverse deviation between subjects' paths. Both subjects in each trial were asked to draw "their" respective route on the map that they were given. The deviation between the respective paths drawn by interlocutors was then determined as the area covered in between the paths (PATHDEV).

Features

Repetition is measured on a lexical and a syntactic level. To do so, we identify all constituents in the utterances as per phrase-structure analysis. *[Go [to [the [[white house] [on [the right]]]]]]* would yield 11 constituents. Each constituent is licensed by a syntactic rule, for instance *VP* $\rightarrow$ *V PP* for the topmost constituent in the above example.

For each constituent, we check whether it is a lexical or syntactic repetition, i.e. if the same words occurred before, or if the licensing rule has occurred before in the same dialogue. If so, we increment counters for lexical and/or syntactic repetitions, and increase a further counter for string repetition by the length of the phrase (in characters). The latter variable accounts for the repetition of long phrases.

We include a data point for each 10-second interval of the dialogue, with features reporting the lexical (LEXREP), syntactic (SYNREP) and character-based (CHARREP) repetitions up to that point in time. A time stamp and the total numbers of constituents and characters are also included (LENGTH). This way, the model may work with repetition proportions rather than the absolute counts.

We train a support vector machine for regression with a radial basis function kernel ($\gamma = 5$), using the features as described above and the PATHDEV score as output.

3.3 Evaluation

We cast the task as a regression problem. To predict a dialogue's score, we apply the SVM to its data points. The mean outcome is the estimated score.

A suitable evaluation measure, the classical r^2, indicates the proportion of the variance in the actual task success score that can be predicted by the model. All results reported here are produced from 10-fold cross-validated 90% training / 10% test splits of the dialogues. No full dialogue was included in both test and training sets.

	Task 1	Task 2
ALL Features	**0.17**	**0.14**
ALL w/o SYNREP	0.15	0.06
ALL w/o LEX/CHARREP	0.09	0.07
LENGTH ONLY	0.09	n/a
Baseline	0.01	0.01

Table 1: Portion of variance explained (r^2)

Task 1 was evaluated with all data, the Task 2 model was trained and tested on data points sampled from the first 5 minutes of the dialogue.

For Task 1 (full dialogues), the results (Table 1) indicate that ALL repetition features together with the LENGTH of the conversation, account for about 17% of the total score variance. The repetition features improve on the performance achieved from dialogue length alone (about 9%).

For the more difficult Task 2, ALL features together achieve 14% of the variance. (Note that LENGTH is not available.) When the syntactic repetition feature is taken out and only lexical (LEXREP) and character repetition (CHARREP) are used, we achieve 6% in explained variance.

The baseline is implemented as a model that always estimates the mean score. It should, theoretically, be close to 0.

3.4 Discussion

Obviously, linguistic information alone will not explain the majority of the task-solving abilities. Apart from subject-related factors, communicative strategies will play a role.

However, linguistic repetition serves as a good predictor of how well interlocutors will complete their joint task. The features used are relatively simple: provided there is some syntactic annotation, rule repetition can easily be detected. Even without syntactic information, lexical repetition already goes a long way.

But what kind of repetition is it that plays a role in task-oriented dialogue? Leaving out features is not an ideal method to quantify their influence–in particular, where features inter-correlate. The contribution of syntactic repetition is still unclear from the

present results: it acts as a useful predictor only over the course of the whole dialogues, but not within a 5-minute time span, where the SVM cannot incorporate its informational content.

We will therefore turn to a more detailed analysis of structural repetition, which should help us draw conclusions relating to the psycholinguistics of dialogue.

4 Long term and short term priming

In the following, we will examine syntactic (structural) priming as one of the driving forces behind alignment. We choose syntactic over lexical priming for two reasons. Lexical repetition due to priming is difficult to distinguish from repetition that is due to interlocutors attending to a particular topic of conversation, which, in coherent dialogue, means that topics are clustered. Lexical choice reflects those topics, hence we expect clusters of particular terminology. Secondly: the maps used to collect the dialogues in the Map Task corpus contained landmarks with labels. It is only natural (even if by means to cross-modal priming) that speakers will identify landmarks using the labels and show little variability in lexical choice. We will measure repetition of syntactic rules, whereby word-by-word repetition (topicality effects, parroting) is explicitly excluded.

For syntactic priming[1], two repetition effects have been identified. Classical priming effects are strong: around 10% for syntactic rules (Reitter et al., 2006b). However, they decay quickly (Branigan et al., 1999) and reach a low plateau after a few seconds, which likens to the effect to semantic (similarity) priming. What complicates matters is that there is also a different, long-term adaptation effect that is also commonly called (repetition) priming.

Adaptation has been shown to last longer, from minutes (Bock and Griffin, 2000) to several days. Lexical boost interactions, where the lexical repetition of material within the repeated structure strengthens structural priming, have been observed for short-term priming, but not for long-term priming trials where material intervened between prime and target utterances (Konopka and Bock, 2005). Thus, short- and long-term adaptation effects may well be due to separate cognitive processes, as recently argued by (Ferreira and Bock, 2006). Section 5 deals with decay-based short-term priming, Section 6 with long-term adaptation.

Pickering and Garrod (2004) do not make the type of priming supporting alignment explicit. Should we find differences in the way task success interacts with different kinds of repetition effects, then this would be a good indication about what effect supports IAM. More concretely, we could say whether alignment is due to the automatic, classical *priming* effect, or whether it is based on a long-term effect that is possibly closer to implicit learning (Chang et al., 2006).

5 Short-term priming

In this section, we attempt to detect differences in the strength of short-term priming in successful and less successful dialogues. To do so, we use the measure of priming strength established by Reitter et al. (2006b), which then allows us to test whether priming interacts with task success. Under the assumptions of IAM we would expect successful dialogues to show more priming than unsuccessful ones.

Obviously, difficulties with the task at hand may be due to a range of problems that the subjects may have, linguistic and otherwise. But given that the dialogues contain variable levels of syntactic priming, one would expect that this has at least some influence on the outcome of the task.

5.1 Method: Logistic Regression

We used mixed-effects regression models that predict a binary outcome (repetition) using a number of discrete and continuous factors.[2]

As a first step, our modeling effort tries to establish a priming effect. To do so, we can make use of the fact that the priming effect decays over time. How strong that decay is gives us an indication of how much repetition probability we see shortly after the stimulus (prime) compared to the probability of chance repetition–without ever explicitly calculating such a prior.

Thus we define the strength of priming as the decay rate of repetition probability, from shortly after

[1] in production and comprehension, which we will not distinguish further for space reasons. Our data are (off-line) production data.

[2] We use Generalized Linear Mixed Effects models fitted using GlmmPQL in the MASS R library.

the prime to 15 seconds afterward (predictor: DIST). Thus, we take several samples at varying distances (d), looking at cases of structural repetition, and cases where structure has not been repeated.

In the syntactic context, syntactic rules such as VP $\rightarrow$ VP PP reflect syntactic decisions. Priming of a syntactic construction shows up in the tendency to repeat such rules in different lexical contexts. Thus, we examine whether syntactic rules have been repeated at a distance d. For each syntactic rule that occurs at time t_1, we check a one-second time period $[t_1 - d - 0.5, t_1 - d + 0.5]$ for an occurrence of the same rule, which would constitute a prime. Thus, the model will be able to implicitly estimate the probability of repetition.

Generalized Linear Regression Models (GLMs) can then model the decay by estimating the relationship between d and the probability of rule repetition. The model is designed to predict whether repetition will occur, or, more precisely, whether there is a prime for a given target (priming). Under a no-priming null-hypothesis, we would assume that the priming probability is independent of d. If there is priming, however, increasing d will negatively influence the priming probability (decay). So, we expect a model parameter (DIST) for d that is reliably negative, and lower, if there is more priming.

With this method, we draw multiple samples from the same utterance–for different d, but also for different syntactic rules occurring in those utterances. Because these samples are inter-dependent, we use a grouping variable indicating the source utterance. Because the dataset is sparse with respect to PRIME, balanced sampling is needed to ensure an equal number of data points of priming and non-priming cases (PRIME) is included.

This method has been previously used to confirm priming effects for the general case of syntactic rules by Reitter et al. (2006b). Additionally, the GLM can take into account categorical and continuous covariates that may interact with the priming effect. In the present experiment, we use an interaction term to model the effect of task success.[3] The crucial interaction, in our case, is task success: PATHDEV is the deviation of the paths that the interlocutors drew, normalized to the range [0,1]. The core model is thus PRIME $\sim log(\text{DIST}) * \text{PATHDEV}$.

If IAM is correct, we would expect that the deviation of paths, which indicates negative task success, will negatively correlate with the priming effect.

5.2 Results

Short-term priming reliably correlated (negatively) with the distance, hence we see a decay and priming effect (DIST, $b = -0.151, p < 0.0001$, as shown in previous work).

Notably, path deviation and short-term priming did not correlate. The model showed was no such interaction (DIST:PATHDEV, $p = 0.91$).

We also tested for an interaction with an additional factor indicating whether prime and target were uttered by the same or a different speaker (comprehension-production vs. production-production priming). No such interaction approached reliability (log(DIST):PATHDEV:ROLE, $p = 0.60$).

We also tested whether priming changes over time over the course of each dialogue. It does not. There were no reliable interaction effects of centered prime/target times (log(DIST):log(STARTTIME), $p = 0.75$, log(DIST):PATHDEV:log(STARTTIME), $p = 0.63$). Reducing the model by removing unreliable interactions did not yield any reliable effects.

5.3 Discussion

We have shown that while there is a clear priming effect in the short term, the size of this priming effect does not correlate with task success. There is no reliable interaction with success.

Does this indicate that there is no strong functional component to priming in the dialogue context? There may still be an influence of cognitive load due to speakers working on the task, or an overall disposition for higher priming in task-oriented dialogue: Reitter et al. (2006b) point at stronger priming in such situations. But our results here are difficult to reconcile with the model suggested by Pickering and Garrod (2004), if we take short-term priming as the driving force behind IAM.

Short-term priming decays within a few seconds. Thus, to what extent could syntactic priming help interlocutors align their situation models? In the Map

[3] We use the $A * B$ operator in the model formulas to indicate the inclusion of main effects of the features A and B and their interactions $A : B$.

Task experiments, interlocutors need to refer to landmarks regularly–but not every few seconds. It would be sensible to expect longer-term adaptation (within minutes) to drive dialogue success.

6 Long-term adapation

Long-term adaptation is a form of priming that occurs over minutes and could, therefore, support linguistic and situation model alignment in task-oriented dialogue. IAM and the success of the SVM based method could be based on such an effect instead of short-term priming. Analogous to the the previous experiment, we hypothesize that more adaptation relates to more task success.

6.1 Method

After the initial few seconds, structural repetition shows little decay, but can be demonstrated even minutes or longer after the stimulus. To measure this type of adapation, we need a different strategy to estimate the size of this effect.

While short-term priming can be pin-pointed using the characteristic decay, for long-term priming we need to inspect whole dialogues and construct and contrast dialogues where priming is possible and ones where it is not. Factor SAMEDOC distinguishes the two situations: 1) Priming can happen in contiguous dialogues. We treat the first half of the dialogue as priming period, and the rule instances in the second half as targets. 2) The control case is when priming cannot have taken place, i.e., between unrelated dialogues. Prime period and targets stem from separate randomly sampled dialogue halves that always come from different dialogues.

Thus, our model (PRIME $\sim$ SAMEDOC $*$ PATHDEV) estimates the influence of priming on rule us. From a Bayesian perspective, we would say that the second kind of data (non-priming) allow the model to estimate a prior for rule repetitions. The goal is now to establish a correlation between SAMEDOC and the existence of repetition. If and only if there is long-term adapation would we expect such a correlation.

Analogous to the short-term priming model, we define repetition as the occurrence of a prime within the first document half (PRIME), and sample rule instances from the second document half. To exclude short-term priming effects, we drop a 10-second portion in the middle of the dialogues.

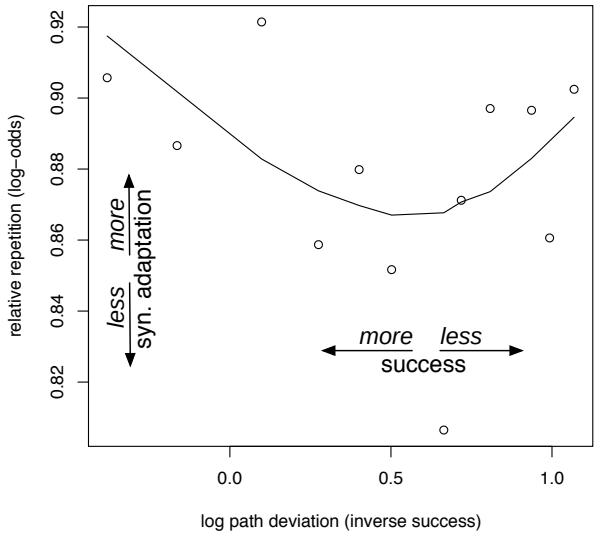


Figure 1: Relative rule repetition probability (chance repetition exluded) over (neg.) task success.

Task success is inverse path deviation PATHDEV as before, which should, under IAM assumptions, interact with the effect estimated for SAMEDOC.

6.2 Results

Long-term repetition showed a positive priming effect (SAMEDOC, $b = 3.303, p < 0.0001$). This generalizes previous experimental priming results in long-term priming.

Long-term-repetition did not interact with (normalized) rule frequency (SAMEDOC:log(RULEFREQ, $b = -0.044, p = 0.35$). The interaction was removed for all other parameters reported.[4]

The effect interacted reliably with the path deviation scores (SAMEDOC:PATHDEV, $b = -0.624, p < 0.05$). We find a reliable correlation of task success and syntactic priming. Stronger path deviations relate to weaker priming.

6.3 Discussion

The more priming we see, the better subjects perform at synchronizing their routes on the maps. This is exactly what one would expect under the assump-

[4]Such an interaction also could not be found in a reduced model with only SAMEDOC and RULEFREQ.

tion of IAM. Also, there is no evidence for stronger long-term adaptation of rare rules, which may point out a qualitative difference to short-term priming.

Of course, this correlation does not necessarily indicate a causal relationship. However, participants in Map Task did not receive an explicit indication about whether they were on the "right track". Mistakes, such as passing a landmark on its East and not on the West side, were made and went unnoticed. Thus, it is not very likely that task success caused alignment to improve at large. We suspect such a possibility, however, for very unsuccessful dialogues. A closer look at the correlation (Figure 1) reveals that while adaptation indeed decreases as task success decreases, adaptation increased again for some of the least successful dialogues. It is possible that here, miscoordination became apparent to the participants, who then tried to switch strategies. Or, simply put: too much alignment (and too little risk-taking) is unhelpful. Further, qualitative, work needs to be done to investigate this hypothesis.

From an applied perspective, the correlation shows that of the repetition effects included in our task-success prediction model, it is long-term syntactic adaptation as opposed to the more automatic short-term priming effect that contributes to prediction accuracy. We take this as an indication to include adaptation rather than just priming in a model of alignment in dialogue.

7 Conclusion

Task success in human-human dialogue is predictable–the more successfully speakers collaborate, the more they show linguistic adaptation. This confirms our initial hypothesis of IAM. In the applied model, knowledge of lexical and syntactic repetition helps to determine task success even after just a few minutes of the conversation.

We suggested two application-oriented tasks (estimating and predicting task success) and an approach to address them. They now provide an opportunity to explore and exploit other linguistic and extra-linguistic parameters.

The second contribution is a closer inspection of structural repetition, which showed that it is long-term adaptation that varies with task success, while short-term priming appears largely autonomous. Long-term adaptation may thus be a strategy that aids dialogue partners in aligning their language and their situation models.

Acknowledgments

The authors would like to thank Frank Keller and the reviewers. The first author is supported by the Edinburgh-Stanford Link.

References

A. Anderson, M. Bader, E. Bard, E. Boyle, G. M. Doherty, S. Garrod, S. Isard, J. Kowtko, J. McAllister, J. Miller, C. Sotillo, H. Thompson, and R. Weinert. 1991. The HCRC Map Task corpus. *Language and Speech*, 34(4):351–366.

J. Kathryn Bock. 1986. Syntactic persistence in language production. *Cognitive Psychology*, 18:355–387.

J. Kathryn Bock and Zenzi Griffin. 2000. The persistence of structural priming: transient activation or implicit learning? *J of Experimental Psychology: General*, 129:177–192.

H. P. Branigan, M. J. Pickering, and A. A. Cleland. 1999. Syntactic priming in language production: evidence for rapid decay. *Psychonomic Bulletin and Review*, 6(4):635–640.

F. Chang, G. Dell, and K. Bock. 2006. Becoming syntactic. *Psychological Review*, 113(2):234–272.

Kenneth W. Church. 2000. Empirial estimates of adaptation: The chance of two noriegas is closer to $p/2$ than p^2. In *Coling-2000*, Saarbrücken, Germany.

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

Vic Ferreira and Kathryn Bock. 2006. The functions of structural priming. *Language and Cognitive Processes*, 21(7-8).

Stefan Th. Gries. 2005. Syntactic priming: A corpus-based approach. *J of Psycholinguistic Research*, 34(4):365–399.

T. Florian Jäger. 2006. *Redundancy and Syntactic Reduction in Spontaneous Speech*. Ph.D. thesis, Stanford University.

Agnieszka Konopka and J. Kathryn Bock. 2005. Helping syntax out: What do words do? In *Proc. 18th CUNY*. Tucson, AZ.

Martin J. Pickering and Holly P. Branigan. 1998. The representation of verbs: Evidence from syntactic priming in language production. *Journal of Memory and Language*, 39:633–651.

Martin J. Pickering and Simon Garrod. 2004. Toward a mechanistic psychology of dialogue. *Behavioral and Brain Sciences*, 27:169–225.

D. Reitter, J. Hockenmaier, and F. Keller. 2006a. Priming effects in Combinatory Categorial Grammar. In *Proc. EMNLP-2006*, pp.308–316. Sydney, Australia.

D. Reitter, J. D. Moore, and F. Keller. 2006b. Priming of syntactic rules in task-oriented dialogue and spontaneous conversation. In *Proc. CogSci-2006*, pp. 685–690. Vancouver, Canada.

Benedikt Szmrecsanyi. 2005. Creatures of habit: A corpus-linguistic analysis of persistence in spoken english. *Corpus Linguistics and Linguistic Theory*, 1(1):113–149.

M. Walker, I. Langkilde, J. Wright, A. Gorin, and D. Litman. 2000. Learning to predict problematic situations in a spoken dialogue system: experiments with How may I help you? In *Proc. NAACL-2000*, pp. 210–217. San Francisco, CA.

Resolving *It, This,* and *That* in Unrestricted Multi-Party Dialog

Christoph Müller
EML Research gGmbH
Villa Bosch
Schloß-Wolfsbrunnenweg 33
69118 Heidelberg, Germany
christoph.mueller@eml-research.de

Abstract

We present an implemented system for the resolution of *it*, *this*, and *that* in transcribed multi-party dialog. The system handles NP-anaphoric as well as discourse-deictic anaphors, i.e. pronouns with VP antecedents. Selectional preferences for NP or VP antecedents are determined on the basis of corpus counts. Our results show that the system performs significantly better than a recency-based baseline.

1 Introduction

This paper describes a fully automatic system for resolving the pronouns *it*, *this*, and *that* in unrestricted multi-party dialog. The system processes manual transcriptions from the ICSI Meeting Corpus (Janin et al., 2003). The following is a short fragment from one of these transcripts. The letters FN in the speaker tag mean that the speaker is a female non-native speaker of English. The brackets and subscript numbers are not part of the original transcript.

FN083: Maybe you can also read through the - all the text which is on the web pages cuz I'd like to change the text a bit cuz sometimes $[\text{it}]_1$'s too long, sometimes $[\text{it}]_2$'s too short, *inbreath* maybe the English is not that good, so *inbreath* um, but anyways - So I tried to do $[\text{this}]_3$ today and if you could do $[\text{it}]_4$ afterwards $[\text{it}]_5$ would be really nice cuz I'm quite sure that I can't find every, like, orthographic mistake in $[\text{it}]_6$ or something. (Bns003)

For each of the six 3rd-person pronouns in the example, the task is to automatically identify its referent, i.e. the entity (if any) to which the speaker makes reference. Once a referent has been identified, the pronoun is resolved by linking it to one of its antecedents, i.e. one of the referent's earlier mentions. For humans, identification of a pronoun's referent is often easy: it_1, it_2, and it_6 are probably used to refer to the text on the web pages, while it_4 is probably used to refer to *reading* this text. Humans also have no problem determining that it_5 is not a normal pronoun at all. In other cases, resolving a pronoun is difficult even for humans: this_3 could be used to refer to either *reading* or *changing* the text on the web pages. The pronoun is ambiguous because evidence for more than one interpretation can be found. Ambiguous pronouns are common in spoken dialog (Poesio & Artstein, 2005), a fact that has to be taken into account when building a spoken dialog pronoun resolution system. Our system is intended as a component in an extractive dialog summarization system. There are several ways in which coreference information can be integrated into extractive summarization. Kabadjov et al. (2005) e.g. obtained their best extraction results by specifying for each sentence whether it contained a mention of a particular anaphoric chain. Apart from improving the extraction itself, coreference information can also be used to substitute anaphors with their antecedents, thus improving the readability of a summary by minimizing the number of dangling anaphors, i.e. anaphors whose antecedents occur in utterances that are not part of the summary. The paper is structured as follows: Section 2 outlines the most important challenges and the state of the art in spoken dialog pronoun resolution. Section 3 describes our annotation experiments, and Section 4 describes the automatic

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 816–823,
Prague, Czech Republic, June 2007.

dialog preprocessing. Resolution experiments and results can be found in Section 5.

2 Pronoun Resolution in Spoken Dialog

Spoken language poses some challenges for pronoun resolution. Some of these arise from nonreferential resp. nonresolvable pronouns, which are important to identify because failure to do so can harm pronoun resolution precision. One common type of nonreferential pronoun is pleonastic *it*. Another cause of nonreferentiality that only applies to spoken language is that the pronoun is *discarded*, i.e. it is part of an incomplete or abandoned utterance. Discarded pronouns occur in utterances that are abandoned altogether.

ME010: Yeah. Yeah. No, no. There was a whole co- There was a little contract signed. It was - Yeah. (Bed017)

If the utterance contains a speech repair (Heeman & Allen, 1999), a pronoun in the *reparandum* part is also treated as discarded because it is not part of the final utterance.

ME10: That's - that's - so that's a - that's a very good question, then - now that it - I understand it. (Bro004)

In the corpus of task-oriented TRAINS dialogs described in Byron (2004), the rate of discarded pronouns is 7 out of 57 (12.3%) for *it* and 7 out of 100 (7.0%) for *that*. Schiffman (1985) reports that in her corpus of career-counseling interviews, 164 out of 838 (19.57%) instances of *it* and 80 out of 582 (13.75%) instances of *that* occur in abandoned utterances.

There is a third class of pronouns which is referential but nonetheless unresolvable: *vague* pronouns (Eckert & Strube, 2000) are characterized by having no clearly defined textual antecedent. Rather, vague pronouns are often used to refer to the topic of the current (sub-)dialog as a whole.

Finally, in spoken language the pronouns *it*, *this*, and *that* are often discourse deictic (Webber, 1991), i.e. they are used to refer to an *abstract object* (Asher, 1993). We treat as abstract objects all referents of VP antecedents, and do not distinguish between VP and S antecedents.

ME013: Well, I mean there's this Cyber Transcriber service, right?

ME025: Yeah, that's true, that's true. (Bmr001)

Discourse deixis is very frequent in spoken dialog: The rate of discourse deictic expressions reported in Eckert & Strube (2000) is 11.8% for pronouns and as much as 70.9% for demonstratives.

2.1 State of the Art

Pronoun resolution in spoken dialog has not received much attention yet, and a major limitation of the few implemented systems is that they are not fully automatic. Instead, they depend on manual removal of unresolvable pronouns like pleonastic *it* and discarded and vague pronouns, which are thus prevented from triggering a resolution attempt. This eliminates a major source of error, but it renders the systems inapplicable in a real-world setting where no such manual preprocessing is feasible.

One of the earliest empirically based works adressing (discourse deictic) pronoun resolution in spoken dialog is Eckert & Strube (2000). The authors outline two algorithms for identifying the antecedents of personal and demonstrative pronouns in two-party telephone conversations from the Switchboard corpus. The algorithms depend on two nontrivial types of information: the incompatibility of a given pronoun with either concrete or abstract antecedents, and the structure of the dialog in terms of dialog acts. The algorithms are not implemented, and Eckert & Strube (2000) report results of the manual application to a set of three dialogs (199 expressions, including other pronouns than *it*, *this*, and *that*). Precision and recall are 66.2 resp. 68.2 for pronouns and 63.6 resp. 70.0 for demonstratives.

An implemented system for resolving personal and demonstrative pronouns in task-oriented TRAINS dialogs is described in Byron (2004). The system uses an explicit representation of domain-dependent semantic category restrictions for predicate argument positions, and achieves a precision of 75.0 and a recall of 65.0 for *it* (50 instances) and a precision of 67.0 and a recall of 62.0 for *that* (93 instances) if all available restrictions are used. Precision drops to 52.0 for *it* and 43.0 for *that* when only domain-independent restrictions are used.

To our knowledge, there is only one implemented system so far that resolves normal and discourse deictic pronouns in unrestricted spoken dialog (Strube & Müller, 2003). The system runs on dialogs from the Switchboard portion of the Penn Treebank. For

it, *this* and *that*, the authors report 40.41 precision and 12.64 recall. The recall does not reflect the actual pronoun resolution performance as it is calculated against all coreferential links in the corpus, not just those with pronominal anaphors. The system draws some non-trivial information from the Penn Treebank, including correct NP chunks, grammatical function tags (subject, object, etc.) and discarded pronouns (based on the `-UNF`-tag). The treebank information is also used for determining the accessibility of potential candidates for discourse deictic pronouns.

In contrast to these approaches, the work described in the following is fully automatic, using only information from the raw, transcribed corpus. No manual preprocessing is performed, so that during testing, the system is exposed to the full range of discarded, pleonastic, and other unresolvable pronouns.

3 Data Collection

The ICSI Meeting Corpus (Janin et al., 2003) is a collection of 75 manually transcribed group discussions of about one hour each, involving three to ten speakers. A considerable number of participants are non-native speakers of English, whose proficiency is sometimes poor, resulting in disfluent or incomprehensible speech. The discussions are real, unstaged meetings on various, technical topics. Most of the discussions are regular weekly meetings of a quite informal conversational style, containing many interrupts, asides, and jokes (Janin, 2002). The corpus features a semi-automatically generated segmentation in which each segment is associated with a speaker tag and a start and end time stamp. Time stamps on the word level are not available. The transcription contains capitalization and punctuation, and it also explicitly records *interruption points* and *word fragments* (Heeman & Allen, 1999), but not the extent of the related disfluencies.

3.1 Annotation

The annotation was done by naive project-external annotators, two non-native and two native speakers of English, with the annotation tool MMAX2[1] on five randomly selected dialogs[2]. The annotation instructions were deliberately kept simple, explaining and illustrating the basic notions of anaphora and discourse deixis, and describing how markables were to be created and linked in the annotation tool. This practice of using a higher number of naive – rather than fewer, highly trained – annotators was motivated by our intention to elicit as many plausible interpretations as possible in the presence of ambiguity. It was inspired by the annotation experiments of Poesio & Artstein (2005) and Artstein & Poesio (2006). Their experiments employed up to 20 annotators, and they allowed for the explicit annotation of ambiguity. In contrast, our annotators were instructed to choose the single most plausible interpretation in case of perceived ambiguity. The annotation covered the pronouns *it*, *this*, and *that* only. Markables for these tokens were created automatically. From among the pronominal[3] instances, the annotators then identified normal, vague, and nonreferential pronouns. For normal pronouns, they also marked the most recent antecedent using the annotation tool's coreference annotation function. Markables for antecedents other than *it*, *this*, and *that* had to be created by the annotators by dragging the mouse over the respective words in the tool's GUI. Nominal antecedents could be either noun phrases (NP) or pronouns (PRO). VP antecedents (for discourse deictic pronouns) spanned only the verb phrase *head*, i.e. the verb, not the entire phrase. By this, we tried to reduce the number of disagreements caused by differing markable demarcations. The annotation of discourse deixis was limited to cases where the antecedent was a finite or infinite verb phrase expressing a proposition, event type, etc.[4]

3.2 Reliability

Inter-annotator agreement was checked by computing the variant of Krippendorff's α described in Passonneau (2004). This metric requires all annotations to contain the same set of markables, a condition that is not met in our case. Therefore, we report α values computed on the *intersection* of the com-

[1] http://mmax.eml-research.de

[2] Bed017, Bmr001, Bns003, Bro004, and Bro005.

[3] The automatically created markables included all instances of *this* and *that*, i.e. also relative pronouns, determiners, complementizers, etc.

[4] Arbitrary spans of text could not serve as antecedents for discourse deictic pronouns. The respective pronouns were to be treated as vague, due to lack of a well-defined antecedent.

pared annotations, i.e. on those markables that can be found in all four annotations. Only a subset of the markables in each annotation is relevant for the determination of inter-annotator agreement: all non-pronominal markables, i.e. all antecedent markables manually created by the annotators, and all referential instances of *it*, *this*, and *that*. The second column in Table 1 contains the cardinality of the union of all four annotators' markables, i.e. the number of all distinct relevant markables in all four annotations. The third and fourth column contain the cardinality and the relative size of the intersection of these four markable sets. The fifth column contains α calculated on the markables in the intersection only. The four annotators only agreed in the identification of markables in approx. 28% of cases. α in the five dialogs ranges from $.43$ to $.52$.

	$\lvert 1 \cup 2 \cup 3 \cup 4 \rvert$	$\lvert 1 \cap 2 \cap 3 \cap 4 \rvert$		α
Bed017	397	109	27.46 %	.47
Bmr001	619	195	31.50 %	.43
Bns003	529	131	24.76 %	.45
Bro004	703	142	20.20 %	.45
Bro005	530	132	24.91 %	.52

Table 1: Krippendorff's α for four annotators.

3.3 Data Subsets

In view of the subjectivity of the annotation task, which is partly reflected in the low agreement even on markable *identification*, the manual creation of a consensus-based gold standard data set did not seem feasible. Instead, we created *core* data sets from all four annotations by means of majority decisions. The core data sets were generated by automatically collecting in each dialog those anaphor-antecedent pairs that at least three annotators identified independently of each other. The rationale for this approach was that an anaphoric link is the more plausible the more annotators identify it. Such a data set certainly contains some spurious or dubious links, while lacking some correct but more difficult ones. However, we argue that it constitutes a plausible subset of anaphoric links that are useful to resolve.

Table 2 shows the number and lengths of anaphoric chains in the core data set, broken down according to the type of the chain-initial antecedent. The rare type OTHER mainly contains adjectival antecedents. More than 75% of all chains consist of two elements only. More than 33% begin with a pronoun. From the perspective of extractive summarization, the resolution of these latter chains is not helpful since there is no non-pronominal antecedent that it can be linked to or substituted with.

	length	2	3	4	5	6	> 6	total
Bed017	NP	17	3	2	-	1	-	23
	PRO	14	-	2	-	-	-	16
	VP	6	1	-	-	-	-	7
	OTHER	-	-	-	-	-	-	-
	all	37 80.44%	4	4	-	1	-	46
Bmr001	NP	14	4	1	1	1	2	23
	PRO	19	9	2	2	1	1	34
	VP	9	5	-	-	-	-	14
	OTHER	-	-	-	-	-	-	-
	all	42 59.16%	18	3	3	2	3	71
Bns003	NP	18	3	3	1	-	-	25
	PRO	18	1	1	-	-	-	20
	VP	14	4	-	-	-	-	18
	OTHER	-	-	-	-	-	-	-
	all	50 79.37%	8	4	1	-	-	63
Bro004	NP	38	5	3	1	-	-	47
	PRO	21	4	-	1	-	-	26
	VP	8	1	1	-	-	-	10
	OTHER	2	1	-	-	-	-	3
	all	69 80.23%	11	4	2	-	-	86
Bro005	NP	37	7	1	-	-	-	45
	PRO	15	3	1	-	-	-	19
	VP	8	1	-	1	-	-	10
	OTHER	3	-	-	-	-	-	3
	all	63 81.82%	11	2	1	-	-	77
Σ	NP	124	22	10	3	2	2	163
	PRO	87	17	6	3	1	1	115
	VP	45	12	1	1	-	-	59
	OTHER	5	1	-	-	-	-	6
	all	261 76.01%	52	17	7	3	3	343

Table 2: Anaphoric chains in core data set.

4 Automatic Preprocessing

Data preprocessing was done fully automatically, using only information from the manual transcription. Punctuation signs and some heuristics were used to split each dialog into a sequence of graphemic sentences. Then, a shallow disfluency detection and removal method was applied, which removed direct repetitions, nonlexicalized filled pauses like *uh*, *um*, interruption points, and word fragments. Each sentence was then matched against a list of potential discourse markers (*actually*, *like*, *you know*, *I mean*, etc.) If a sentence contained one or more matches, string variants were created in which the respective words were deleted. Each of these variants was then submitted to a parser trained on written text (Charniak, 2000). The variant with the highest probability (as determined by the parser) was chosen. NP chunk markables were created for all non-recursive NP constituents identi-

fied by the parser. Then, VP chunk markables were created. Complex verbal constructions like MD + INFINITIVE were modelled by creating markables for the individual expressions, and attaching them to each other with labelled relations like INFINITIVE_COMP. NP chunks were also attached, using relations like SUBJECT, OBJECT, etc.

5 Automatic Pronoun Resolution

We model pronoun resolution as binary classification, i.e. as the mapping of anaphoric mentions to previous mentions of the same referent. This method is not incremental, i.e. it cannot take into account earlier resolution decisions or any other information beyond that which is conveyed by the two mentions. Since more than 75% of the anaphoric chains in our data set would not benefit from incremental processing because they contain one anaphor only, we see this limitation as acceptable. In addition, incremental processing bears the risk of system degradation due to error propagation.

5.1 Features

In the binary classification model, a pronoun is resolved by creating a set of candidate antecedents and searching this set for a matching one. This search process is mainly influenced by two factors: exclusion of candidates due to *constraints*, and selection of candidates due to *preferences* (Mitkov, 2002). Our features encode information relevant to these two factors, plus more generally descriptive factors like distance etc. Computation of all features was fully automatic.

Shallow constraints for nominal antecedents include number, gender and person incompatibility, embedding of the anaphor into the antecedent, and coargumenthood (i.e. the antecedent and anaphor must not be governed by the same verb). For VP antecedents, a common shallow constraint is that the anaphor must not be governed by the VP antecedent (so-called *argumenthood*). Preferences, on the other hand, define conditions under which a candidate probably is the correct antecedent for a given pronoun. A common shallow preference for nominal antecedents is the parallel function preference, which states that a pronoun with a particular grammatical function (i.e. subject or object) preferably has an antecedent with a similar function. The subject preference, in contrast, states that subject antecedents are generally preferred over those with less salient functions, independent of the grammatical function of the anaphor. Some of our features encode this functional and structural parallelism, including identity of form (for PRO antecedents) and identity of grammatical function or governing verb.

A more sophisticated constraint on NP antecedents is what Eckert & Strube (2000) call *I-Incompatibility*, i.e. the semantic incompatibility of a pronoun with an individual (i.e. NP) antecedent. As Eckert & Strube (2000) note, subject pronouns in copula constructions with adjectives that can only modify abstract entities (like e.g. *true*, *correct*, *right*) are incompatible with concrete antecedents like *car*. We postulate that the preference of an adjective to modify an abstract entity (in the sense of Eckert & Strube (2000)) can be operationalized as the conditional probability of the adjective to appear with a *to*-infinitive resp. a *that*-sentence complement, and introduce two features which calculate the respective preference on the basis of corpus[5] counts. For the first feature, the following query is used:

$$\frac{\text{\# it (’s|is|was|were) ADJ to}}{\text{\# it (’s|is|was|were) ADJ}}$$

According to Eckert & Strube (2000), pronouns that are objects of verbs which mainly take sentence complements (like *assume*, *say*) exhibit a similar incompatibility with NP antecedents, and we capture this with a similar feature. Constraints for VPs include the following: VPs are inaccessible for discourse deictic reference if they fail to meet the *right frontier* condition (Webber, 1991). We use a feature which is similar to that used by Strube & Müller (2003) in that it approximates the *right frontier* on the basis of syntactic (rather than discourse structural) relations. Another constraint is *A-Incompatibility*, i.e. the incompatibility of a pronoun with an abstract (i.e. VP) antecedent. According to Eckert & Strube (2000), subject pronouns in copula constructions with adjectives that can only modify concrete entities (like e.g. *expensive*, *tasty*) are incompatible with abstract antecedents, i.e. they

[5] Based on the approx. 250,000,000 word TIPSTER corpus (Harman & Liberman, 1994).

cannot be discourse deictic. The function of this constraint is already covered by the two corpus-based features described above in the context of *I-Incompatibility*. Another feature, based on Yang et al. (2005), encodes the semantic compatibility of anaphor and NP antecedent. We operationalize the concept of semantic compatibility by substituting the anaphor with the antecedent head and performing corpus queries. E.g., if the anaphor is object, the following query[6] is used:

$$\frac{\#\,(\text{V}|\text{Vs}|\text{Ved}|\text{Ving})\ (\varnothing|\text{a}|\text{an}|\text{the}|\text{this}|\text{that})\ \text{ANTE}+}{\#\,(\text{V}|\text{Vs}|\text{Ved}|\text{Ving})\ (\varnothing|\text{the}|\text{these}|\text{those})\ \text{ANTES}}$$
$$\#\,(\text{ANTE}|\text{ANTES})$$

If the anaphor is the subject in an adjective copula construction, we use the following corpus count to quantify the compatibility between the predicated adjective and the NP antecedent (Lapata et al., 1999):

$$\frac{\#\ \text{ADJ}\ (\text{ANTE}|\text{ANTES}) + \#\ \text{ANTE}\ (\text{is}|\text{was})\ \text{ADJ}+}{\#\ \text{ANTES}\ (\text{are}|\text{were})\ \text{ADJ}}$$
$$\#\ \text{ADJ}$$

A third class of more general properties of the potential anaphor-antecedent pair includes the type of anaphor (personal vs. demonstrative), type of antecedent (definite vs. indefinite noun phrase, pronoun, finite vs. infinite verb phrase, etc.). Special features for the identification of discarded expressions include the distance (in words) to the closest preceeding resp. following disfluency (indicated in the transcription as an interruption point, word fragment, or *uh* resp. *um*). The relation between potential anaphor and (any type of) antecedent is described in terms of distance in seconds[7] and words. For VP antecedents, the distance is calculated from the *last* word in the entire phrase, not from the phrase *head*. Another feature which is relevant for dialog encodes whether both expressions are uttered by the same speaker.

[6] V is the verb governing the anaphor. Correct inflected forms were also generated for irregular verbs. ANTE resp. ANTES is the singular resp. plural head of the antecedent.

[7] Since the data does not contain word-level time stamps, this distance is determined on the basis of a simple forced alignment. For this, we estimated the number of syllables in each word on the basis of its vowel clusters, and simply distributed the known duration of the segment evenly on all words it contains.

5.2 Data Representation and Generation

Machine learning data for training and testing was created by pairing each anaphor with each of its compatible potential antecedents within a certain temporal distance (9 seconds for NP and 7 seconds for VP antecedents), and labelling the resulting data instance as *positive* resp. *negative*. VP antecedent candidates were created only if the anaphor was either *that*[8] or the object of a form of *do*.

Our core data set does not contain any nonreferential pronouns, though the classifier is exposed to the full range of pronouns, including discarded and otherwise nonreferential ones, during testing. We try to make the classifier robust against nonreferential pronouns in the following way: From the manual annotations, we select instances of *it*, *this*, and *that* that at least three annotators identified as nonreferential. For each of these, we add the full range of all-negative instances to the training data, applying the constraints mentioned above.

5.3 Evaluation Measure

As Bagga & Baldwin (1998) point out, in an application-oriented setting, not all anaphoric links are equally important: If a pronoun is resolved to an anaphoric chain that contains only pronouns, this resolution can be treated as neutral because it has no application-level effect. The common coreference evaluation measure described in Vilain et al. (1995) is inappropriate in this setting. We calculate precision, recall and F-measure on the basis of the following definitions: A pronoun is resolved correctly resp. incorrectly only if it is linked (directly or transitively) to the correct resp. incorrect *non-pronominal* antecedent. Likewise, the number of maximally resolvable pronouns in the core data set (i.e. the evaluation *key*) is determined by considering only pronouns in those chains that do not begin with a pronoun. Note that our definition of precision is stricter (and yields lower figures) than that applied in the ACE context, as the latter ignores incorrect links between two expressions in the *response*

[8] It is a common observation that demonstratives (in particular *that*) are preferred over *it* for discourse deictic reference (Schiffman, 1985; Webber, 1991; Asher, 1993; Eckert & Strube, 2000; Byron, 2004; Poesio & Artstein, 2005). This preference can also be observed in our core data set: 44 out of 59 VP antecedents (69.49%) are anaphorically referred to by *that*.

if these expressions happen to be unannotated in the *key*, while we treat them as precision errors unless the antecedent is a pronoun. The same is true for links in the *response* that were identified by less than three annotators in the *key*. While it is practical to treat those links as wrong, it is also simplistic because it does not do justice to ambiguous pronouns (cf. Section 6).

5.4 Experiments and Results

Our best machine learning results were obtained with the Weka[9] Logistic Regression classifier.[10] All experiments were performed with dialog-wise cross-validation. For each run, training data was created from the manually annotated markables in four dialogs from the core data set, while testing was performed on the automatically detected chunks in the remaining fifth dialog. For training and testing, the person, number[11], gender, and (co-)argument constraints were used. If an anaphor gave rise to a positive instance, no negative training instances were created beyond that instance. If a referential anaphor did not give rise to a positive training instance (because its antecedent fell outside the search scope or because it was removed by a constraint), no instances were created for that anaphor. Instances for nonreferential pronouns were added to the training data as described in Section 5.2.

During testing, we select for each potential anaphor the positive antecedent with the highest overall confidence. Testing parameters include `it-filter`, which switches on and off the module for the detection of nonreferential *it* described in Müller (2006). When evaluated alone, this module yields a precision of 80.0 and a recall of 60.9 for the detection of pleonastic and discarded *it* in the five ICSI dialogs. For training, this module was always on. We also vary the parameter `tipster`, which controls whether or not the corpus frequency features are used. If `tipster` is off, we ignore the corpus frequency features both during training and testing.

We first ran a simple baseline system which resolved pronouns to their most recent compatible antecedent, applying the same settings and constraints as for testing (cf. above). The results can be found in the first part of Table 3. Precision, recall and F-measure are provided for ALL and for NP and VP antecedents individually. The parameter `tipster` is not available for the baseline system. The best baseline performance is precision 4.88, recall 20.06 and F-measure 7.85 in the setting with `it-filter` on. As expected, this filter yields an increase in precision and a decrease in recall. The negative effect is outweighed by the positive effect, leading to a small but insignificant[12] increase in F-measure for all types of antecedents.

Setting		Ante	Baseline P	Baseline R	Baseline F	Logistic Regression P	Logistic Regression R	Logistic Regression F
-it-filter	-tipster	NP	4.62	27.12	7.90	18.53	20.34	19.39*
		VP	1.72	2.63	2.08	13.79	10.53	11.94
		ALL	4.40	20.69	7.25	17.67	17.56	17.61*
	+tipster	NP	-	-	-	19.33	22.03	20.59***
		VP	-	-	-	13.43	11.84	12.59
		ALL	-	-	-	18.16	19.12	18.63**
+it-filter	-tipster	NP	5.18	26.27	8.65	17.87	17.80	17.83*
		VP	1.77	2.63	2.12	13.12	10.53	11.68
		ALL	4.88	20.06	7.85	16.89	15.67	16.26*
	+tipster	NP	-	-	-	20.82	21.61	21.21**
		VP	-	-	-	11.27	10.53	10.88
		ALL	-	-	-	18.67	18.50	18.58**

Table 3: Resolution results.

The second part of Table 3 shows the results of the Logistic Regression classifier. When compared to the best baseline, the F-measures are consistently better for NP, VP, and ALL. The improvement is (sometimes highly) significant for NP and ALL, but never for VP. The best F-measure for ALL is 18.63, yielded by the setting with `it-filter` off and `tipster` on. This setting also yields the best F-measure for VP and the second best for NP. The contribution of the it-filter is disappointing: In both `tipster` settings, the it-filter causes F-measure for ALL to go down. The contribution of the corpus features, on the other hand, is somewhat inconclusive: In both `it-filter` settings, they cause an increase in F-measure for ALL. In the first setting, this increase is accompanied by an increase in F-measure for VP, while in the second setting, F-measure for VP goes down. It has to be noted, however, that none of the improvements brought about by the it-filter or the tipster corpus features is statistically significant. This also confirms some of the findings of Kehler et al. (2004), who found features similar to

[9] http://www.cs.waikato.ac.nz/ml/weka/

[10] The full set of experiments is described in Müller (2007).

[11] The *number* constraint applies to *it* only, as *this* and *that* can have both singular and plural antecedents (Byron, 2004).

[12] Significance of improvement in F-measure is tested using a paired one-tailed t-test and $p <= 0.05$ (*), $p <= 0.01$ (**), and $p <= 0.005$ (***).

our tipster corpus features not to be significant for NP-anaphoric pronoun resolution in written text.

6 Conclusions and Future Work

The system described in this paper is – to our knowledge – the first attempt towards fully automatic resolution of NP-anaphoric and discourse deictic pronouns (*it*, *this*, and *that*) in multi-party dialog. Unlike other implemented systems, it is usable in a realistic setting because it does not depend on manual pronoun preselection or non-trivial discourse structure or domain knowledge. The downside is that, at least in our strict evaluation scheme, the performance is rather low, especially when compared to that of state-of-the-art systems for pronoun resolution in written text. In future work, it might be worthwhile to consider less rigorous and thus more appropriate evaluation schemes in which links are weighted according to how many annotators identified them.

In its current state, the system only processes manual dialog transcripts, but it also needs to be evaluated on the output of an automatic speech recognizer. While this will add more noise, it will also give access to useful prosodic features like stress.

Finally, the system also needs to be evaluated extrinsically, i.e. with respect to its contribution to dialog summarization. It might turn out that our system already has a positive effect on extractive summarization, even though its performance is low in absolute terms.

Acknowledgments. This work has been funded by the Deutsche Forschungsgemeinschaft as part of the DIANA-Summ project (STR-545/2-1,2) and by the Klaus Tschira Foundation. We are grateful to the anonymous ACL reviewers for helpful comments and suggestions. We also thank Ron Artstein for help with significance testing.

References

Artstein, R. & M. Poesio (2006). Identifying reference to abstract objects in dialogue. In *Proc. of BranDial-06*, pp. 56–63.

Asher, N. (1993). *Reference to Abstract Objects in Discourse.* Dordrecht, The Netherlands: Kluwer.

Bagga, A. & B. Baldwin (1998). Algorithms for scoring coreference chains. In *Proc. of LREC-98*, pp. 79–85.

Byron, D. K. (2004). *Resolving pronominal reference to abstract entities.*, (Ph.D. thesis). University of Rochester.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proc. of NAACL-00*, pp. 132–139.

Eckert, M. & M. Strube (2000). Dialogue acts, synchronising units and anaphora resolution. *Journal of Semantics*, 17(1):51–89.

Harman, D. & M. Liberman (1994). *TIPSTER Complete LDC93T3A.* 3 CD-ROMS. Linguistic Data Consortium, Philadelphia, Penn., USA.

Heeman, P. & J. Allen (1999). Speech repairs, intonational phrases, and discourse markers: Modeling speakers' utterances in spoken dialogue. *Computational Linguistics*, 25(4):527–571.

Janin, A. (2002). Meeting recorder. In *Proceedings of the Applied Voice Input/Output Society Conference (AVIOS)*, San Jose, California, USA, May 2002.

Janin, A., D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke & C. Wooters (2003). The ICSI Meeting Corpus. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Hong Kong, pp. 364–367.

Kabadjov, M. A., M. Poesio & J. Steinberger (2005). Task-based evaluation of anaphora resolution: The case of summarization. In *Proceedings of the RANLP Workshop on Crossing Barriers in Text Summarization Research*, Borovets, Bulgaria.

Kehler, A., D. Appelt, L. Taylor & A. Simma (2004). The (non)utility of predicate-argument frequencies for pronoun interpretation. In *Proc. of HLT-NAACL-04*, pp. 289–296.

Lapata, M., S. McDonald & F. Keller (1999). Determinants of adjective-noun plausibility. In *Proc. of EACL-99*, pp. 30–36.

Mitkov, R. (2002). *Anaphora Resolution.* London, UK: Longman.

Müller, C. (2006). Automatic detection of nonreferential it in spoken multi-party dialog. In *Proc. of EACL-06*, pp. 49–56.

Müller, C. (2007). *Fully automatic resolution of* it, this, *and* that *in unrestricted multi-party dialog.*, (Ph.D. thesis). Eberhard Karls Universität Tübingen, Germany. To appear.

Passonneau, R. J. (2004). Computing reliability for coreference annotation. In *Proc. of LREC-04*.

Poesio, M. & R. Artstein (2005). The reliability of anaphoric annotation, reconsidered: Taking ambiguity into account. In *Proceedings of the ACL Workshop on Frontiers in Corpus Annotation II: Pie in the Sky*, pp. 76–83.

Schiffman, R. J. (1985). *Discourse constraints on 'it' and 'that': A Study of Language Use in Career Counseling Interviews.*, (Ph.D. thesis). University of Chicago.

Strube, M. & C. Müller (2003). A machine learning approach to pronoun resolution in spoken dialogue. In *Proc. of ACL-03*, pp. 168–175.

Vilain, M., J. Burger, J. Aberdeen, D. Connolly & L. Hirschman (1995). A model-theoretic coreference scoring scheme. In *Proc. of MUC-6*, pp. 45–52.

Webber, B. L. (1991). Structure and ostension in the interpretation of discourse deixis. *Language and Cognitive Processes*, 6(2):107–135.

Yang, X., J. Su & C. L. Tan (2005). Improving pronoun resolution using statistics-based semantic compatibility information. In *Proc. of ACL-05*, pp. 165–172.

A Comparative Study of Parameter Estimation Methods for Statistical Natural Language Processing

Jianfeng Gao[*], Galen Andrew[*], Mark Johnson[*&], Kristina Toutanova[*]
[*]Microsoft Research, Redmond WA 98052, {jfgao,galena,kristout}@microsoft.com
[&]Brown University, Providence, RI 02912, mj@cs.brown.edu

Abstract

This paper presents a comparative study of five parameter estimation algorithms on four NLP tasks. Three of the five algorithms are well-known in the computational linguistics community: Maximum Entropy (ME) estimation with L_2 regularization, the Averaged Perceptron (AP), and Boosting. We also investigate ME estimation with L_1 regularization using a novel optimization algorithm, and BLasso, which is a version of Boosting with Lasso (L_1) regularization. We first investigate all of our estimators on two re-ranking tasks: a parse selection task and a language model (LM) adaptation task. Then we apply the best of these estimators to two additional tasks involving conditional sequence models: a Conditional Markov Model (CMM) for part of speech tagging and a Conditional Random Field (CRF) for Chinese word segmentation. Our experiments show that across tasks, three of the estimators — ME estimation with L_1 or L_2 regularization, and AP — are in a near statistical tie for first place.

1 Introduction

Parameter estimation is fundamental to many statistical approaches to NLP. Because of the high-dimensional nature of natural language, it is often easy to generate an extremely large number of features. The challenge of parameter estimation is to find a combination of the typically noisy, redundant features that accurately predicts the target output variable and avoids overfitting. Intuitively, this can be achieved either by selecting a small number of highly-effective features and ignoring the others, or by averaging over a large number of weakly informative features. The first intuition motivates feature selection methods such as Boosting and BLasso (e.g., Collins 2000; Zhao and Yu, 2004), which usually work best when many features are completely irrelevant. L_1 or Lasso regularization of linear models, introduced by Tibshirani (1996), embeds feature selection into regularization so that both an assessment of the reliability of a feature and the decision about whether to remove it are done in the same framework, and has generated a large amount of interest in the NLP community recently (e.g., Goodman 2003; Riezler and Vasserman 2004). If on the other hand most features are noisy but at least weakly correlated with the target, it may be reasonable to attempt to reduce noise by averaging over all of the features. ME estimators with L_2 regularization, which have been widely used in NLP tasks (e.g., Chen and Rosenfeld 2000; Charniak and Johnson 2005; Johnson et al. 1999), tend to produce models that have this property. In addition, the perceptron algorithm and its variants, e.g., the voted or averaged perceptron, is becoming increasingly popular due to their competitive performance, simplicity in implementation and low computational cost in training (e.g., Collins 2002).

While recent studies claim advantages for L_1 regularization, this study is the first of which we are aware to systematically compare it to a range of estimators on a diverse set of NLP tasks. Gao et al. (2006) showed that BLasso, due to its explicit use of L_1 regularization, outperformed Boosting in the LM adaptation task. Ng (2004) showed that for logistic regression, L_1 regularization outperforms L_2 regularization on artificial datasets which contain many completely irrelevant features. Goodman (2003) showed that in two out of three tasks, an ME estimator with a one-sided Laplacian prior (i.e., L_1 regularization with the constraint that all feature weights are positive) outperformed a comparable estimator using a Gaussian prior (i.e., L_2 regularization). Riezler and Vasserman (2004) showed that an L_1-regularized ME estimator outperformed an L_2-regularized estimator for ranking the parses of a stochastic unification-based grammar.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 824–831,
Prague, Czech Republic, June 2007.

While these individual estimators are well described in the literature, little is known about the relative performance of these methods because the published results are generally not directly comparable. For example, in the parse re-ranking task, one cannot tell whether the L_2- regularized ME approach used by Charniak and Johnson (2005) significantly outperforms the Boosting method by Collins (2000) because different feature sets and *n*-best parses were used in the evaluations of these methods.

This paper conducts a much-needed comparative study of these five parameter estimation algorithms on four NLP tasks: ME estimation with L_1 and L_2 regularization, the Averaged Perceptron (AP), Boosting, and BLasso, a version of Boosting with Lasso (L_1) regularization. We first investigate all of our estimators on two re-ranking tasks: a parse selection task and a language model adaptation task. Then we apply the best of these estimators to two additional tasks involving conditional sequence models: a CMM for POS tagging and a CRF for Chinese word segmentation. Our results show that ME estimation with L_2 regularization achieves the best performing estimators in all of the tasks, and AP achieves almost as well and requires much less training time. L_1 (Lasso) regularization also performs well and leads to sparser models.

2 Estimators

All the four NLP tasks studied in this paper are based on linear models (Collins 2000) which require learning a mapping from inputs $x \in X$ to outputs $y \in Y$. We are given:

- Training samples (x_i, y_i) for $i = 1 \ldots N$,
- A procedure $\boldsymbol{GEN}$ to generate a set of candidates $\boldsymbol{GEN}(x)$ for an input *x*,
- A feature mapping $\Phi: X \times Y \mapsto \mathbb{R}^D$ to map each (x, y) to a vector of feature values, and
- A parameter vector $\boldsymbol{w} \in \mathbb{R}^D$, which assigns a real-valued weight to each feature.

For all models except the CMM sequence model for POS tagging, the components $\boldsymbol{GEN}$, Φ and $\boldsymbol{w}$ directly define a mapping from an input x to an output $F(x)$ as follows:

$$F(x) = \arg\max_{y \in \boldsymbol{GEN}(X)} \Phi(x, y) \cdot \boldsymbol{w}. \quad (1)$$

In the CMM sequence classifier, locally normalized linear models to predict the tag of each word token are chained together to arrive at a probability estimate for the entire tag sequence, resulting in a slightly different decision rule.

Linear models, though simple, can capture very complex dependencies because the features can be arbitrary functions of the input/output pair. For example, we can define a feature to be the log conditional probability of the output as estimated by some other model, which may in turn depend on arbitrarily complex interactions of 'basic' features. In practice, with an appropriate feature set, linear models achieve very good empirical results on various NLP tasks. The focus of this paper however is not on feature definition (which requires domain knowledge and varies from task to task), but on parameter estimation (which is generic across tasks). We assume we are given fixed feature templates from which a large number of features are generated. The task of the estimator is to use the training samples to choose a parameter vector $\boldsymbol{w}$, such that the mapping $F(x)$ is capable of correctly classifying unseen examples. We will describe the five estimators in our study individually.

2.1 ME estimation with L_2 regularization

Like many linear models, the ME estimator chooses $\boldsymbol{w}$ to minimize the sum of the empirical loss on the training set and a regularization term:

$$\widehat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \{L(\boldsymbol{w}) + R(\boldsymbol{w})\}. \quad (2)$$

In this case, the loss term $L(\mathbf{w})$ is the negative conditional log-likelihood of the training data,

$$L(\boldsymbol{w}) = -\sum_{i=1}^{n} \log P(y_i|x_i), \text{ where}$$

$$P(y|x) = \frac{\exp(\Phi(x, y) \cdot \boldsymbol{w})}{\sum_{y' \in GEN(x)} \exp(\Phi(x, y') \cdot \boldsymbol{w})}$$

and the regularizer term $R(\boldsymbol{w}) = \alpha \sum_j w_j^2$ is the weighted squared L_2 norm of the parameters. Here, α is a parameter that controls the amount of regularization, optimized on held-out data.

This is one of the most popular estimators, largely due to its appealing computational properties: both $L(\boldsymbol{w})$ and $R(\boldsymbol{w})$ are convex and differentiable, so gradient-based numerical algorithms can be used to find the global minimum efficiently.

In our experiments, we used the *limited memory quasi-Newton* algorithm (or L-BFGS, Nocedal and Wright 1999) to find the optimal $\boldsymbol{w}$ because this method has been shown to be substantially faster than other methods such as Generalized Iterative Scaling (Malouf 2002).

Because for some sentences there are multiple best parses (i.e., parses with the same F-Score), we used the variant of ME estimator described in Riezler et al. (2002), where $L(\boldsymbol{w})$ is defined as the likelihood of the best parses $y \in Y(x)$ relative to the *n*-best parser output $\boldsymbol{GEN}(x)$, (i.e., $Y(x) \sqsubseteq \boldsymbol{GEN}(x)$): $L(\boldsymbol{w}) = -\sum_{i=1}^{n} \log \sum_{y_i \in Y(x_i)} P(y_i|x_i)$.

We applied this variant in our experiments of parse re-ranking and LM adaptation, and found that on both tasks it leads to a significant improvement in performance for the L_2-regularied ME estimator but not for the L_1-regularied ME estimator.

2.2 ME estimation with L_1 regularization

This estimator also minimizes the negative conditional log-likelihood, but uses an L_1 (or Lasso) penalty. That is, $R(\boldsymbol{w})$ in Equation (2) is defined according to $R(\boldsymbol{w}) = \alpha \sum_j |w_j|$. L_1 regularization typically leads to sparse solutions in which many feature weights are exactly zero, so it is a natural candidate when feature selection is desirable. By contrast, L_2 regularization produces solutions in which most weights are small but non-zero.

Optimizing the L_1-regularized objective function is challenging because its gradient is discontinuous whenever some parameter equals zero. Kazama and Tsujii (2003) described an estimation method that constructs an equivalent constrained optimization problem with twice the number of variables. However, we found that this method is impractically slow for large-scale NLP tasks. In this work we use the *orthant-wise limited-memory quasi-Newton* algorithm (OWL-QN), which is a modification of L-BFGS that allows it to effectively handle the discontinuity of the gradient (Andrew and Gao 2007). We provide here a high-level description of the algorithm.

A quasi-Newton method such as L-BFGS uses first order information at each iterate to build an approximation to the Hessian matrix, $\boldsymbol{H}$, thus modeling the local curvature of the function. At each step, a search direction is chosen by minimizing a quadratic approximation to the function:

$$Q(x) = \frac{1}{2}(x - x_0)'\boldsymbol{H}(x - x_0) + g_0'(x - x_0)$$

where x_0 is the current iterate, and g_0 is the function gradient at x_0. If $\boldsymbol{H}$ is positive definite, the minimizing value of x can be computed analytically according to: $x^* = x_0 - \boldsymbol{H}^{-1} g_0$.

L-BFGS maintains vectors of the change in gradient $g_k - g_{k-1}$ from the most recent iterations, and uses them to construct an estimate of the inverse Hessian $\boldsymbol{H}^{-1}$. Furthermore, it does so in such a way that $\boldsymbol{H}^{-1} g_0$ can be computed without expanding out the full matrix, which is typically unmanageably large. The computation requires a number of operations linear in the number of variables.

OWL-QN is based on the observation that when restricted to a single orthant, the L_1 regularizer *is* differentiable, and is in fact a linear function of $\boldsymbol{w}$. Thus, so long as each coordinate of any two consecutive search points does not pass through zero, $R(\boldsymbol{w})$ does not contribute at all to the curvature of the function on the segment joining them. Therefore, we can use L-BFGS to approximate the Hessian of $L(\boldsymbol{w})$ alone, and use it to build an approximation to the full regularized objective that is valid on a given orthant. To ensure that the next point is in the valid region, we project each point during the line search back onto the chosen orthant.[1] At each iteration, we choose the orthant containing the current point and into which the direction giving the greatest local rate of function decrease points.

This algorithm, although only a simple modification of L-BFGS, works quite well in practice. It typically reaches convergence in even fewer iterations than standard L-BFGS takes on the analogous L_2-regularized objective (which translates to less training time, since the time per iteration is only negligibly higher, and total time is dominated by function evaluations). We describe OWL-QN more fully in (Andrew and Gao 2007). We also show that it is significantly faster than Kazama and Tsujii's algorithm for L_1 regularization and prove that it is guaranteed converge to a parameter vector that globally optimizes the L_1-regularized objective.

2.3 Boosting

The Boosting algorithm we used is based on Collins (2000). It optimizes the pairwise *exponential loss* (ExpLoss) function (rather than the logarithmic loss optimized by ME). Given a training sample (x_i, y_i), for each possible output $y_j \in \boldsymbol{GEN}(x_i)$, we

[1] This projection just entails zeroing-out any coordinates that change sign. Note that it is possible for a variable to change sign in two iterations, by moving from a negative value to zero, and on a the next iteration moving from zero to a positive value.

1 Set w_0 = argmin$_{w0}$ExpLoss(w); and w_d = 0 for d=1…D
2 Select a feature f_{k^*} which has largest estimated impact on reducing ExpLoss of Equation (3)
3 Update $\lambda_{k^*} \leftarrow \lambda_{k^*} + \delta^*$, and return to Step 2

Figure 1: The boosting algorithm

define the margin of the pair (y_i, y_j) with respect to $\boldsymbol{w}$ as $M(y_i, y_j) = \Phi(x_i, y_i) \cdot \boldsymbol{w} - \Phi(x_i, y_j) \cdot \boldsymbol{w}$.

Then ExpLoss is defined as

$$\text{ExpLoss}(\boldsymbol{w}) = \sum_i \sum_{y_j \in \boldsymbol{GEN}(x_i)} \exp\left(-\text{M}(\text{y}_\text{i}, \text{y}_\text{j})\right) \quad (3)$$

Figure 1 summarizes the Boosting algorithm we used. It is an incremental feature selection procedure. After initialization, Steps 2 and 3 are repeated T times; at each iteration, a feature is chosen and its weight is updated as follows.

First, we define $\text{Upd}(\boldsymbol{w}, k, \delta)$ as an updated model, with the same parameter values as w with the exception of w_k, which is incremented by δ:

$$\text{Upd}(\boldsymbol{w}, k, \delta) = (w_1, \ldots, w_k + \delta, \ldots, w_D)$$

Then, Steps 2 and 3 in Figure 1 can be rewritten as Equations (4) and (5), respectively.

$$(k^*, \delta^*) = \arg\min_{k,\delta} \text{ExpLoss}(\text{Upd}(\boldsymbol{w}, k, \delta)) \quad (4)$$

$$\boldsymbol{w}^t = \text{Upd}(\boldsymbol{w}^{t-1}, k^*, \delta^*) \quad (5)$$

Because Boosting can overfit we update the weight of f_{k^*} by a small fixed step size ε, as in Equation (6), following the FSLR algorithm (Hastie et al. 2001).

$$\boldsymbol{w}^t = \text{Upd}(\boldsymbol{w}^{t-1}, k^*, \epsilon \times \text{sign}(\delta^*)) \quad (6)$$

By taking such small steps, Boosting imposes a kind of implicit regularization, and can closely approximate the effect of L_1 regularization in a local sense (Hastie et al. 2001). Empirically, smaller values of ϵ lead to smaller numbers of test errors.

2.4 Boosted Lasso

The Boosted Lasso (BLasso) algorithm was originally proposed in Zhao and Yu (2004), and was adapted for language modeling by Gao et al. (2006). BLasso can be viewed as a version of Boosting with L_1 regularization. It optimizes an L_1-regularized ExpLoss function:

$$\text{LassoLoss}(\boldsymbol{w}) = \text{ExpLoss}(\boldsymbol{w}) + R(\boldsymbol{w}) \quad (7)$$

where $R(\boldsymbol{w}) = \alpha \sum_j |w_j|$.

BLasso also uses an incremental feature selection procedure to learn parameter vector $\boldsymbol{w}$, just as Boosting does. Due to the explicit use of the regularization term $R(\boldsymbol{w})$, however, there are two major differences from Boosting.

At each iteration, BLasso takes either a *forward step* or a *backward step*. Similar to Boosting, at each forward step, a feature is selected and its weight is updated according to Eq. (8) and (9).

$$(k^*, \delta^*) = arg \min_{k,\delta=\pm\epsilon} \text{ExpLoss}(\text{Upd}(\boldsymbol{w}, k, \delta)) \quad (8)$$

$$\boldsymbol{w}^t = \text{Upd}(\boldsymbol{w}^{t-1}, k^*, \epsilon \times \text{sign}(\delta^*)) \quad (9)$$

There is a small but important difference between Equations (8) and (4). In Boosting, as shown in Equation (4), a feature is selected by its impact on reducing the loss with its optimal update δ^*. By contrast, in BLasso, as shown in Equation (8), rather than optimizing over δ for each feature, the loss is calculated with an update of either $+\epsilon$ or $-\epsilon$, i.e., grid search is used for feature weight estimation. We found in our experiments that this modification brings a consistent improvement.

The backward step is unique to BLasso. At each iteration, a feature is selected and the absolute value of its weight is reduced by ϵ if and only if it leads to a decrease of the LassoLoss, as shown in Equations (10) and (11), where θ is a tolerance parameter.

$$k^* = \arg\min_{k:w_k \neq 0} \text{ExpLoss}(\text{Upd}(\boldsymbol{w}, k, -\epsilon\text{sign}(w_k)) \quad (10)$$

$$\boldsymbol{w}^t = \text{Upd}(\boldsymbol{w}^{t-1}, k^*, \text{sign}(w_{k^*}) \times \epsilon) \quad (11)$$

$$\text{if } \text{LassoLoss}(\boldsymbol{w}^{t-1}, \alpha^{t-1}) - \text{LassoLoss}(\boldsymbol{w}^t, \alpha^t) > \theta$$

Figure 2 summarizes the BLasso algorithm we used. After initialization, Steps 4 and 5 are repeated T times; at each iteration, a feature is chosen and its weight is updated either backward or forward by a fixed amount ϵ. Notice that the value of α is adaptively chosen according to the reduction of ExpLoss during training. The algorithm starts with a large initial α, and then at each forward step the value of α decreases until ExpLoss stops decreasing. This is intuitively desirable: it is expected that most highly effective features are selected in early stages of training, so the reduction of ExpLoss at each step in early stages are more substantial than in later stages. These early steps coincide with the Boosting steps most of the time. In other words, the effect of backward steps is more visible at later stages. It can be proved that for a finite number of features and θ=0, the BLasso algorithm shown in Figure 2 converges to the Lasso solution when $\epsilon \to 0$. See Gao et al. (2006) for implementation details, and Zhao and Yu (2004) for a theoretical justification for BLasso.

1 Initialize w^0: set $w_0 = \text{argmin}_{w0}\text{ExpLoss}(w)$, and $w_d = 0$ for d=1…D.
2 Take a forward step according to Eq. (8) and (9), and the updated model is denoted by w^1
3 Initialize $\alpha = (\text{ExpLoss}(w^0)-\text{ExpLoss}(w^1))/\varepsilon$
4 Take a backward step if and only if it leads to a decrease of LassoLoss according to Eq. (10) and (11), where $\theta = 0$; otherwise
5 Take a forward step according to Eq. (8) and (9); update $\alpha = \min(\alpha, (\text{ExpLoss}(w^{t-1})-\text{ExpLoss}(w^t))/\varepsilon)$; and return to Step 4.

Figure 2: The BLasso algorithm

1 Set $w_0 = 1$ and $w_d = 0$ for d=1…D
2 For t = 1…T (T = the total number of iterations)
3 For each training sample (x_i, y_i), i = 1…N
4 Choose the best candidate z_i from GEN(x_i) using the current model w,

$$z_i = \arg\max_{z \in GEN(x\_i)} \Phi(x_i, z) \cdot w$$

5 w = w + $\eta(\Phi(x_i, y_i) - \Phi(x_i, z_i))$, where η is the size of learning step, optimized on held-out data.

Figure 3: The perceptron algorithm

2.5 Averaged Perceptron

The perceptron algorithm can be viewed as a form of incremental training procedure (e.g., using stochastic approximation) that optimizes a *minimum square error* (MSE) loss function (Mitchell, 1997). As shown in Figure 3**,** it starts with an initial parameter setting and updates it for each training example. In our experiments, we used the Averaged Perceptron algorithm of Freund and Schapire (1999), a variation that has been shown to be more effective than the standard algorithm (Collins 2002). Let $\boldsymbol{w}^{t,i}$ be the parameter vector after the i^{th} training sample has been processed in pass t over the training data. The average parameters are defined as $\bar{\boldsymbol{w}} = \frac{1}{TN}\sum_t \sum_i \boldsymbol{w}^{t,i}$ where T is the number of epochs, and N is the number of training samples.

3 Evaluations

From the four tasks we consider, parsing and language model adaptation are both examples of re-ranking. In these tasks, we assume that we have been given a list of candidates $\boldsymbol{GEN}(x)$ for each training or test sample (x, y), generated using a *baseline* model. Then, a linear model of the form in Equation (1) is used to discriminatively re-rank the candidate list using additional features which may or may not be included in the baseline model. Since the mapping from x to y by the linear model may make use of arbitrary global features of the output and is performed "all at once", we call such a linear model a *global model*.

In the other two tasks (i.e., Chinese word segmentation and POS tagging), there is no explicit enumeration of $\boldsymbol{GEN}(x)$. The mapping from x to y is determined by a sequence model which aggregates the decisions of local linear models via a dynamic program. In the CMM, the local linear models are trained independently, while in the CRF model, the local models are trained jointly. We call these two linear models *local models* because they dynamically combine the output of models that use only local features.

While it is straightforward to apply the five estimators to global models in the re-ranking framework, the application of some estimators to the local models is problematic. Boosting and BLasso are too computationally expensive to be applied to CRF training and we compared the other three better performing estimation methods for this model. The CMM is a probabilistic sequence model and the log-loss used by ME estimation is most natural for it; thus we limit the comparison to the two kinds of ME models for CMMs. Note that our goal is not to compare locally trained models to globally trained ones; for a study which focuses on this issue, see (Punyakanok et al. 2005).

In each task we compared the performance of different estimators using task-specific measures. We used the Wilcoxon signed rank test to test the statistical significance of the difference among the competing estimators. We also report other results such as number of non-zero features after estimation, number of training iterations, and computation time (in minutes of elapsed time on an XEON™ MP 3.6GHz machine).

3.1 Parse re-ranking

We follow the experimental paradigm of parse re-ranking outlined in Charniak and Johnson (2005), and fed the features extracted by their program to the five rerankers we developed. Each uses a linear model trained using one of the five estimators. These rerankers attempt to select the best parse y for a sentence x from the 50-best list of possible parses $\boldsymbol{GEN}(x)$ for the sentence. The linear model combines the log probability calculated by the Charniak (2000) parser as a feature with 1,219,272 additional features. We trained the fea-

	F-Score	# features	time (min)	# train iter
Baseline	0.8986			
ME/L2	0.9176	1,211,026	62	129
ME/L1	0.9165	19,121	37	174
AP	0.9164	939,248	2	8
Boosting	0.9131	6,714	495	92,600
BLasso	0.9133	8,085	239	56,500

Table 1: Performance summary of estimators on parsing re-ranking (ME/L2: ME with L_2 regularization; ME/L1: ME with L_1 regularization)

	ME/L2	ME/L1	AP	Boost	BLasso
ME/L2		>>	~	>>	>>
ME/L1	<<		~	>	~
AP	~	~		>>	>
Boost	<<	<	<<		~
Blasso	<<	~	<	~	

Table 2: Statistical significance test results (">>" or "<<" means P-value < 0.01; > or < means 0.01 < P-value ≤ 0.05; "~" means P-value > 0.05)

ture weights **w** on Sections 2-19 of the Penn Treebank, adjusted the regularizer constant α to maximize the F-Score on Sections 20-21 of the Treebank, and evaluated the rerankers on Section 22. The results are presented in Tables 1[2] and 2, where **Baseline** results were obtained using the parser by Charniak (2000).

The ME estimation with L_2 regularization outperforms all of the other estimators significantly except for the AP, which performs almost as well and requires an order of magnitude less time in training. Boosting and BLasso are feature selection methods in nature, so they achieve the sparsest models, but at the cost of slightly lower performance and much longer training time. The L_1-regularized ME estimator also produces a relatively sparse solution whereas the Averaged Perceptron and the L_2-regularized ME estimator assign almost all features a non-zero weight.

3.2 Language model adaptation

Our experiments with LM adaptation are based on the work described in Gao et al. (2006). The variously trained language models were evaluated according to their impact on Japanese text input accuracy, where input phonetic symbols x are mapped into a word string y. Performance of the application is measured in terms of character error rate (CER), which is the number of characters wrongly converted from x divided by the number of characters in the correct transcript.

	CER	# features	time (min)	#train iter
Baseline	10.24%			
MAP	7.98%			
ME/L2	6.99%	295,337	27	665
ME/L1	7.01%	53,342	25	864
AP	7.23%	167,591	6	56
Boost	7.54%	32,994	175	71,000
BLasso	7.20%	33,126	238	250,000

Table 3. Performance summary of estimators (lower is better) on language model adaptation

	ME/L2	ME/L1	AP	Boost	BLasso
ME/L2		~	>>	>>	>>
ME/L1	~		>>	>>	>>
AP	<<	<<		>>	~
Boost	<<	<<	<<		<<
BLasso	<<	<<	~	>>	

Table 4. Statistical significance test results.

Again we evaluated five linear rerankers, one for each estimator. These rerankers attempt to select the best conversions y for an input phonetic string x from a 100-best list $\boldsymbol{GEN}(x)$ of possible conversions proposed by a baseline system. The linear model combines the log probability under a trigram language model as base feature and additional 865,190 word uni/bi-gram features. These uni/bi-gram features were already included in the trigram model which was trained on a *background* domain corpus (Nikkei Newspaper). But in the linear model their feature weights were trained discriminatively on an *adaptation* domain corpus (Encarta Encyclopedia). Thus, this forms a cross domain adaptation paradigm. This also implies that the portion of redundant features in this task could be much larger than that in the parse re-ranking task, especially because the background domain is reasonably similar to the adaptation domain.

We divided the Encarta corpus into three sets that do not overlap. A 72K-sentences set was used as training data, a 5K-sentence set as development data, and another 5K-sentence set as testing data. The results are presented in Tables 3 and 4, where **Baseline** is the word-based trigram model trained on background domain corpus, and **MAP** (maximum *a posteriori*) is a traditional model adaptation method, where the parameters of the background model are adjusted so as to maximize the likelihood of the adaptation data.

[2] The result of ME/L2 is better than that reported in Andrew and Gao (2007) due to the use of the variant of L_2-regularized ME estimator, as described in Section 2.1.

	Test F_1	# features	# train iter
ME/L2	0.9719	8,084,086	713
ME/L1	0.9713	317,146	201
AP	0.9703	1,965,719	162

Table 5. Performance summary of estimators on CWS

The results are more or less similar to those in the parsing task with one visible difference: L_1 regularization achieved relatively better performance in this task. For example, while in the parsing task ME with L_2 regularization significantly outperforms ME with L_1 regularization, their performance difference is not significant in this task. While in the parsing task the performance difference between BLasso and Boosting is not significant, BLasso outperforms Boosting significantly in this task. Considering that a much higher proportion of the features are redundant in this task than the parsing task, the results seem to corroborate the observation that L_1 regularization is robust to the presence of many redundant features.

3.3 Chinese word segmentation

Our third task is Chinese word segmentation (CWS). The goal of CWS is to determine the boundaries between words in a section of Chinese text. The model we used is the hybrid Markov/semi- Markov CRF described by Andrew (2006), which was shown to have state-of-the-art accuracy. We tested models trained with the various estimation methods on the Microsoft Research Asia corpus from the Second International Chinese Word Segmentation, and we used the same train/test split used in the competition. The model and experimental setup is identical with that of Andrew (2006) except for two differences. First, we extracted features from both positive and negative training examples, while Andrew (2006) uses only features that occur in some positive training example. Second, we used the last 4K sentences of the training data to select the weight of the regularizers and to determine when to stop perceptron training.

We compared three of the best performing estimation procedures on this task: ME with L_2 regularization, ME with L_1 regularization, and the Averaged Perceptron. In this case, ME refers to minimizing the negative log-probability of the correct segmentation, which is globally normalized, while the perceptron is trained using at each iteration the exact maximum-scoring segmentation with the current weights. We observed the same pattern as in the other tasks: the three algorithms have nearly identical performance, while L_1 uses only 6% of the features, and the Averaged Perceptron requires significantly fewer training iterations. In this case, L_1 was also several times faster than L_2. The results are summarized in Table 5.[3]

We note that all three algorithms performed slightly better than the model used by Andrew (2006), which also used L_2 regularization (96.84 F_1). We believe the difference is due to the use of features derived from negative training examples.

3.4 POS tagging

Finally we studied the impact of the regularization methods on a Maximum Entropy conditional Markov Model (MEMM, McCallum et al. 2000) for POS tagging. MEMMs decompose the conditional probability of a tag sequence given a word sequence as follows:

$$P(t_1 \dots t_n | w_1 \dots w_n) = \prod_{i=1}^{n} P(t_i | t_{i-1} \dots t_{i-k}, w_1 \dots w_n)$$

where the probability distributions for each tag given its context are ME models. Following previous work (Ratnaparkhi, 1996), we assume that the tag of a word is independent of the tags of all preceding words given the tags of the previous two words (i.e., $k=2$ in the equation above). The local models at each position include features of the current word, the previous word, the next word, and features of the previous two tags. In addition to lexical identity of the words, we used features of word suffixes, capitalization, and number/special character signatures of the words.

We used the standard splits of the Penn Treebank from the tagging literature (Toutanova et al. 2003) for training, development and test sets. The training set comprises Sections 0-18, the development set — Sections 19-21, and the test set — Sections 22-24. We compared training the ME models using L_1 and L_2 regularization. For each of the two types of regularization we selected the best value of the regularization constant using grid search to optimize the accuracy on the development set. We report final accuracy measures on the test set in Table 6.

The results on this task confirm the trends we have seen so far. There is almost no difference in

[3] Only the L2 vs. AP comparison is significant at a 0.05 level according to the Wilcoxon signed rank test.

	Accuracy (%)	# features	# train iter
MEMM/L2	96.39	926,350	467
MEMM/L1	96.41	84,070	85

Table 6. Performance summary of estimators on POS tagging

accuracy of the two kinds of regularizations, and indeed the differences were not statistically significant. Estimation with L_1 regularization required considerably less time than estimation with L_2, and resulted in a model which is more than ten times smaller.

4 Conclusions

We compared five of the most competitive parameter estimation methods on four NLP tasks employing a variety of models, and the results were remarkably consistent across tasks. Three of the methods — ME estimation with L_2 regularization, ME estimation with L_1 regularization, and the Averaged Perceptron — were nearly indistinguishable in terms of test set accuracy, with ME estimation with L_2 regularization perhaps enjoying a slight lead. Meanwhile, ME estimation with L_1 regularization achieves the same level of performance while at the same time producing sparse models, and the Averaged Perceptron provides an excellent compromise of high performance and fast training.

These results suggest that when deciding which type of parameter estimation to use on these or similar NLP tasks, one may choose any of these three popular methods and expect to achieve comparable performance. The choice of which to implement should come down to other considerations: if model sparsity is desired, choose ME estimation with L_1 regularization (or feature selection methods such as BLasso); if quick implementation and training is necessary, use the Averaged Perceptron; and ME estimation with L_2 regularization may be used if it is important to achieve the highest obtainable level of performance.

References

Andrew, G. 2006. A hybrid Markov/semi-Markov conditional random field for sequence segmentation. In *EMNLP*, 465-472.

Andrew, G. and Gao, J. 2007. Scalable training of L_1-regularized log-linear models. In *ICML*.

Charniak, E. 2000. A maximum-entropy-inspired parser. In *NAACL*, 132-139.

Charniak, E. and Johnson, M. 2005. Coarse-to-fine *n*-best parsing and MaxEnt discriminative re-ranking. In *ACL*. 173-180.

Chen, S.F., and Rosenfeld, R. 2000. A survey of smoothing techniques for ME models. *IEEE Trans. On Speech and Audio Processing*, 8(2): 37-50.

Collins, M. 2000. Discriminative re-ranking for natural language parsing. In *ICML*, 175-182.

Collins, M. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *EMNLP*, 1-8.

Freund, Y, R. Iyer, R. E. Schapire, and Y. Singer. 1998. An efficient boosting algorithm for combining preferences. In *ICML'98*.

Freund, Y. and Schapire, R. E. 1999. Large margin classification using the perceptron algorithm. In *Machine Learning*, 37(3): 277-296.

Hastie, T., R. Tibshirani and J. Friedman. 2001. *The elements of statistical learning*. Springer-Verlag, New York.

Gao, J., Suzuki, H., and Yu, B. 2006. Approximation lasso methods for language modeling. In *ACL*.

Goodman, J. 2004. Exponential priors for maximum entropy models. In *NAACL*.

Johnson, M., Geman, S., Canon, S., Chi, Z., and Riezler, S. 1999. Estimators for stochastic "Unification-based" grammars. In *ACL*.

Kazama, J. and Tsujii, J. 2003. Evaluation and extension of maximum entropy models with inequality constraints. In *EMNLP*.

Malouf, R. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *HLT*.

McCallum A, D. Freitag and F. Pereira. 2000. Maximum entropy markov models for information extraction and segmentation. In *ICML*.

Mitchell, T. M. 1997. *Machine learning*. The McGraw-Hill Companies, Inc.

Ng, A. Y. 2004. Feature selection, L_1 vs. L_2 regularization, and rotational invariance. In *ICML*.

Nocedal, J., and Wright, S. J. 1999. *Numerical Optimization*. Springer, New York.

Punyakanok, V., D. Roth, W. Yih, and D. Zimak. 2005. Learning and inference over constrained output. In *IJCAI*.

Ratnaparkhi, A. 1996. A maximum entropy part-of-speech tagger. In EMNLP.

Riezler, S., and Vasserman, A. 2004. Incremental feature selection and L_1 regularization for relax maximum entropy modeling. In *EMNLP*.

Riezler, S., King, T. H., Kaplan, R. M., Crouch, R., Maxwell, J., and Johnson, M. 2002. Parsing the wall street journal using a lexical-functional grammar and discriminative estimation techniques. In *ACL*. 271-278.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *J. R. Statist. Soc. B*, 58(1): 267-288.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. 2003. Feature-rich Part-of-Speech tagging with a cyclic dependency network. In *HLT-NAACL*, 252-259.

Zhao, P. and B. Yu. 2004. Boosted lasso. *Tech Report*, Statistics Department, U. C. Berkeley.

Grammar Approximation by Representative Sublanguage: A New Model for Language Learning

Smaranda Muresan
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742, USA
smara@umiacs.umd.edu

Owen Rambow
Center for Computational Learning Systems
Columbia University
New York, NY 10027, USA
rambow@cs.columbia.edu

Abstract

We propose a new language learning model that learns a syntactic-semantic grammar from a small number of natural language strings annotated with their semantics, along with basic assumptions about natural language syntax. We show that the search space for grammar induction is a complete grammar lattice, which guarantees the uniqueness of the learned grammar.

1 Introduction

There is considerable interest in learning computational grammars.[1] While much attention has focused on learning syntactic grammars either in a supervised or unsupervised manner, recently there is a growing interest toward learning grammars/parsers that capture semantics as well (Bos et al., 2004; Zettlemoyer and Collins, 2005; Ge and Mooney, 2005).

Learning both syntax and semantics is arguably more difficult than learning syntax alone. In formal grammar learning theory it has been shown that learning from "good examples," or representative examples, is more powerful than learning from all the examples (Freivalds et al., 1993). Haghighi and Klein (2006) show that using a handful of "prototypes" significantly improves over a fully unsupervised PCFG induction model (their prototypes were formed by sequences of POS tags; for example, prototypical NPs were DT NN, JJ NN).

In this paper, we present a new grammar formalism and a new learning method which together address the problem of learning a syntactic-semantic grammar in the presence of a representative sample of strings annotated with their semantics, along with minimal assumptions about syntax (such as syntactic categories). The semantic representation is an ontology-based semantic representation. The annotation of the representative examples does not include the entire derivation, unlike most of the existing syntactic treebanks. The aim of the paper is to present the formal aspects of our grammar induction model.

In Section 2, we present a new grammar formalism, called *Lexicalized Well-Founded Grammars*, a type of constraint-based grammars that combine syntax and semantics. We then turn to the two main results of this paper. In Section 3 we show that our grammars can always be learned from a set of positive representative examples (with no negative examples), and the search space for grammar induction is a complete grammar lattice, which guarantees the uniqueness of the learned grammar. In Section 4, we propose a new computationally efficient model for grammar induction from pairs of utterances and their semantic representations, called *Grammar Approximation by Representative Sublanguage (GARS)*. Section 5 discusses the practical use of our model and Section 6 states our conclusions and future work.

[1]This research was supported by the National Science Foundation under Digital Library Initiative Phase II Grant Number IIS-98-17434 (Judith Klavans and Kathleen McKeown, PIs). We would like to thank Judith Klavans for her contributions over the course of this research, Kathy McKeown for her input, and several anonymous reviewers for very useful feedback on earlier drafts of this paper.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 832–839,
Prague, Czech Republic, June 2007.

2 Lexicalized Well-Founded Grammars

Lexicalized Well-Founded Grammars (LWFGs) are a type of Definite Clause Grammars (Pereira and Warren, 1980) where: (1) the Context-Free Grammar backbone is extended by introducing a partial ordering relation among nonterminals (well-founded) 2) each string is associated with a syntactic-semantic representation called *semantic molecule*; 3) grammar rules have two types of constraints: one for semantic composition and one for ontology-based semantic interpretation.

The partial ordering among nonterminals allows the ordering of the grammar rules, and thus facilitates the bottom-up induction of these grammars.

The *semantic molecule* is a syntactic-semantic representation of natural language strings $w' = \binom{h}{b}$ where h *(head)* encodes the information required for semantic composition, and b *(body)* is the actual semantic representation of the string. Figure 1 shows examples of semantic molecules for an adjective, a noun and a noun phrase. The representations associated with the lexical items are called *elementary semantic molecules* (I), while the representations built by the combination of others are called *derived semantic molecules* (II). The head of the semantic molecule is a flat feature structure, having at least two attributes encoding the syntactic category of the associated string, cat, and the head of the string, head. The set of attributes is finite and known a priori for each syntactic category. The body of the semantic molecule is a flat, ontology-based semantic representation. It is a logical form, built as a conjunction of atomic predicates $\langle concept \rangle.\langle attr \rangle = \langle concept \rangle$, where variables are either concept or slot identifiers in an ontology. For example, the adjective *major* is represented as $\langle X_1.isa = major, X_2.Y = X_1 \rangle$, which says that the meaning of an adjective is a concept ($X_1.isa = major$), which is a value of a property of another concept ($X_2.Y = X_1$) in the ontology.

The grammar nonterminals are augmented with pairs of strings and their semantic molecules. These pairs are called syntagmas, and are denoted by $\sigma = (w, w') = (w, \binom{h}{b})$. There are two types of constraints at the grammar rule level — one for *semantic composition* (defines how the meaning of a natural language expression is composed from the meaning of its parts) and one for *ontology-based semantic interpretation*. An example of a LWFG rule is given in Figure 1(III). The composition constraints Φ_{comp} applied to the heads of the semantic molecules, form a system of equations that is a simplified version of "path equations" (Shieber et al., 1983), because the heads are flat feature structures. These constraints are learned together with the grammar rules. The ontology-based constraints represent the validation on the ontology, and are applied to the body of the semantic molecule associated with the left-hand side nonterminal. They are not learned. Currently, Φ_{onto} is a predicate which can succeed or fail. When it succeeds, it instantiates the variables of the semantic representation with concepts/slots in the ontology. For example, given the phrase *major damage*, Φ_{onto} succeeds and returns (X_1=MAJOR, X=DAMAGE, Y=DEGREE), while given the phrase *major birth* it fails. We leave the discussion of the ontology constraints for a future paper, since it is not needed for the main result of this paper.

We give below the formal definition of Lexical-

I. Elementary Semantic Molecules

(major/adj)′ = $\begin{pmatrix} \begin{bmatrix} \text{cat} & \text{adj} \\ \text{head} & X_1 \\ \text{mod} & X_2 \end{bmatrix} \\ h_1 \\ b_1 \left\langle X_1.\text{isa} = \text{major},\ X_2.\text{Y}=X_1 \right\rangle \end{pmatrix}$

(damage/noun)′ = $\begin{pmatrix} \begin{bmatrix} \text{cat} & \text{noun} \\ \text{nr} & \text{sg} \\ \text{head} & X_3 \end{bmatrix} \\ h_3 \\ b_3 \left\langle X_3.\text{isa} = \text{damage} \right\rangle \end{pmatrix}$

II. Derived Semantic Molecule

(major damage)′ = $\begin{pmatrix} \begin{bmatrix} \text{cat} & \text{n} \\ \text{nr} & \text{sg} \\ \text{head} & \text{X} \end{bmatrix} \\ h \\ b \left\langle X_1.\text{isa} = \text{major},\ \text{X.Y}=X_1,\ \text{X.isa}=\text{damage} \right\rangle \end{pmatrix}$

III. Constraint Grammar Rule

$N(w, \binom{h}{b}) \rightarrow Adj(w_1, \binom{h_1}{b_1}), N(w_2, \binom{h_2}{b_2}) : \Phi_{comp}, \Phi_{onto}(b)$

$\Phi_{comp}(h, h_1, h_2) = \{h.cat = n, h.head = h_1.mod,$
$h.head = h_2.head, h.nr = h_2.nr, h_1.cat = adj, h_2.cat = n\}$

$\Phi_{onto}(b)$ returns X_1=MAJOR, X=DAMAGE, Y=DEGREE from ontology

Figure 1: Examples of two elementary semantic molecules (I), a derived semantic molecule (II) obtained by combining them, and a constraint grammar rule together with the constraints Φ_{comp}, Φ_{onto} (III) .

ized Well-Founded Grammars, except that we do not define formally the constraints due to lack of space (see (Muresan, 2006) for details).

Definition 1. A *Lexicalized Well-Founded Grammar (LWFG)* is a 6-tuple, $G = \langle \Sigma, \Sigma', N_G, \succeq, P_G, S \rangle$, where:

1. Σ is a finite set of terminal symbols.
2. Σ' is a finite set of elementary semantic molecules corresponding to the set of terminal symbols.
3. N_G is a finite set of nonterminal symbols.
4. $\succeq$ is a partial ordering relation among the nonterminals.
5. P_G is a set of constraint rules. A constraint rule is written $A(\sigma) \rightarrow B_1(\sigma_1), \ldots, B_n(\sigma_n)\colon \Phi(\bar{\sigma})$, where $\bar{\sigma} = (\sigma, \sigma_1, ..., \sigma_n)$ such that $\sigma = (w, w'), \sigma_i = (w_i, w_i'), 1 \leq i \leq n, w = w_1 \cdots w_n, w' = w_1' \circ \cdots \circ w_n'$, and $\circ$ is the semantic composition operator. For brevity, we denote a rule by $A \rightarrow \beta\colon \Phi$, where $A \in N_G, \beta \in N_G^+$. For the rules whose left-hand side are preterminals, $A(\sigma) \rightarrow$, we use the notation $A \rightarrow \sigma$. There are three types of rules: ordered non-recursive, ordered recursive, and non-ordered rules. A grammar rule $A(\sigma) \rightarrow B_1(\sigma_1), \ldots, B_n(\sigma_n)\colon \Phi(\bar{\sigma})$, is an *ordered rule*, if for all B_i, we have $A \succeq B_i$. In LWFGs, each nonterminal symbol is a left-hand side in at least one ordered non-recursive rule and the empty string cannot be derived from any nonterminal symbol.
6. $S \in N_G$ is the start nonterminal symbol, and $\forall A \in N_G, S \succeq A$ (we use the same notation for the reflexive, transitive closure of $\succeq$).

The relation $\succeq$ is a partial ordering only among nonterminals, and it should not be confused with information ordering derived from the flat feature structures. This relation makes the set of nonterminals well-founded, which allows the ordering of the grammar rules, as well as the ordering of the syntagmas generated by LWFGs.

Definition 2. Given a LWFG, G, the *ground syntagma derivation* relation, $\stackrel{*G}{\Rightarrow}$,[2] is defined as: $\frac{A \rightarrow \sigma}{A \stackrel{*G}{\Rightarrow} \sigma}$ (if $\sigma = (w, w'), w \in \Sigma, w' \in \Sigma'$, i.e., A is a preterminal), and $\frac{B_i \stackrel{*G}{\Rightarrow} \sigma_i,\ i=1,\ldots,n, \quad A(\sigma) \rightarrow B_1(\sigma_1), \ldots, B_n(\sigma_n)\colon \Phi(\bar{\sigma})}{A \stackrel{*G}{\Rightarrow} \sigma}$.

In LWFGs all syntagmas $\sigma = (w, w')$, derived from a nonterminal A have the same category of their semantic molecules w'.[3]

The language of a grammar G is the set of all syntagmas generated from the start symbol S, i.e., $L(G) = \{\sigma | \sigma = (w, w'), w \in \Sigma^+, S \stackrel{*G}{\Rightarrow} \sigma\}$. The *set of all syntagmas* generated by *a grammar* G is $L_\sigma(G) = \{\sigma | \sigma = (w, w'), w \in \Sigma^+, \exists A \in N_G, A \stackrel{*G}{\Rightarrow} \sigma\}$. Given a LWFG G we call a set $E_\sigma \subseteq L_\sigma(G)$ a sublanguage of G. Extending the notation, given a LWFG G, the set of syntagmas generated by a *rule* $(A \rightarrow \beta\colon \Phi) \in P_G$ is $L_\sigma(A \rightarrow \beta\colon \Phi) = \{\sigma | \sigma = (w, w'), w \in \Sigma^+, (A \rightarrow \beta\colon \Phi) \stackrel{*G}{\Rightarrow} \sigma\}$, where $(A \rightarrow \beta\colon \Phi) \stackrel{*G}{\Rightarrow} \sigma$ denotes the ground derivation $A \stackrel{*G}{\Rightarrow} \sigma$ obtained using the rule $A \rightarrow \beta\colon \Phi$ in the last derivation step (we have bottom-up derivation). We will use the short notation $L_\sigma(r)$, where r is a grammar rule.

Given a LWFG G and a sublanguage E_σ (not necessarily of G) we denote by $\mathbb{S}(G) = L_\sigma(G) \cap E_\sigma$, the set of syntagmas generated by G *reduced to the sublanguage* E_σ. Given a grammar rule $r \in P_G$, we call $\mathbb{S}(r) = L_\sigma(r) \cap E_\sigma$ the set of syntagmas generated by r *reduced to the sublanguage* E_σ.

As we have previously mentioned, the partial ordering among grammar nonterminals allows the ordering of the syntagmas generated by the grammar, which allows us to define the *representative examples* of a LWFG.

Representative Examples. Informally, the representative examples E_R of a LWFG, G, are the simplest syntagmas ground-derived by the grammar G, i.e., for each grammar rule there exist a syntagma which is ground-derived from it in the minimum number of steps. Thus, the size of the representative example set is equal with the size of the set of grammar rules, $|E_R| = |P_G|$.

This set of representative examples is used by the grammar learning model to generate the candidate hypotheses. For generalization, a larger sublanguage $E_\sigma \supseteq E_R$ is used, which we call representative sublanguage.

[2] The ground derivation ("reduction" in (Wintner, 1999)) can be viewed as the bottom-up counterpart of the usual derivation.

[3] This property is used for determining the lhs nonterminal of the learned rule.

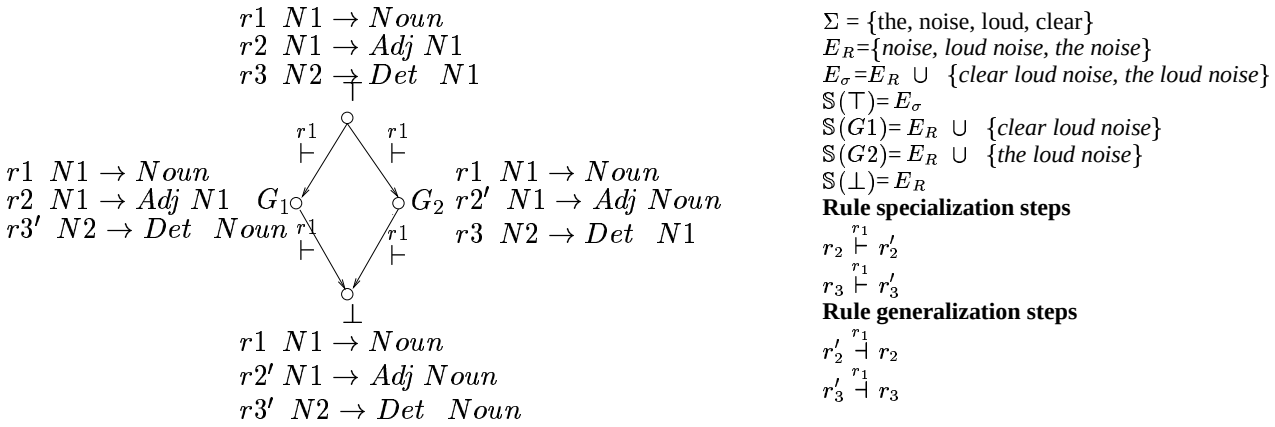


Figure 2: Example of a simple grammar lattice. All grammars generate E_R, and only $\top$ generates E_σ (Σ is a common lexicon for all the grammars)

3 A Grammar Lattice as a Search Space for Grammar Induction

In this section we present a class of Lexicalized Well-Founded Grammars that form a complete lattice. This grammar lattice is the search space for our grammar induction model, which we present in Section 4. An example of a grammar lattice is given in Figure 2, where for simplicity, we only show the context-free backbone of the grammar rules, and only strings, not syntagmas. Intuitively, the grammars found lower in the lattice are more specialized than the ones higher in the lattice. For learning, E_R is used to generate the most specific hypotheses (grammar rules), and thus all the grammars should be able to generate those examples. The sublanguage E_σ is used during generalization, thus only the most general grammar, $\top$, is able to generate the entire sublanguage. In other words, the generalization process is bounded by E_σ, that is why our model is called Grammar Approximation by Representative Sublanguage.

There are two properties that LWFGs should have in order to form a complete lattice: 1) they should be unambiguous, and 2) they should preserve the parsing of the representative example set, E_R. We define these two properties in turn.

Definition 3. A LWFG, G, is *unambiguous* w.r.t. a sublanguage $E_\sigma \subseteq L_\sigma(G)$ if $\forall \sigma \in E_\sigma$ there is one and only one rule that derives σ.

Since the unambiguity is relative to a set of syntagmas (pairs of strings and their semantic molecules) and not to a set of natural language strings, the requirement is compatible with modeling natural language. For example, an ambiguous string such as *John saw the man with the telescope* corresponds to two unambiguous syntagmas.

In order to define the second property, we need to define *the rule specialization step* and *the rule generalization step* of unambiguous LWFGs, such that they are *E_R-parsing-preserving* and are the inverse of each other. The property of E_R-parsing-preserving means that both the initial and the specialized/generalized rules ground-derive the same syntagma, $\sigma_A \in E_R$.

Definition 4. The *rule specialization step*:

$$\frac{A(\sigma_A)\rightarrow\alpha B(\sigma_B^*)\gamma\colon \Phi_A \qquad B(\sigma_B)\rightarrow\beta\colon \Phi_B}{A(\sigma_A)\rightarrow\alpha\beta\gamma\colon \Phi'_A}$$

is *E_R-parsing-preserving*, if there exists $\sigma_A \in E_R$ and $r_{gen} \overset{*G}{\Rightarrow} \sigma_A$ and $r_{spec} \overset{*G'}{\Rightarrow} \sigma_A$, where $r_{gen} = A(\sigma_A) \rightarrow \alpha B(\sigma_B^*)\gamma\colon \Phi_A$, $r_B = B(\sigma_B) \rightarrow \beta\colon \Phi_B$, and $r_{spec} = A(\sigma_A) \rightarrow \alpha\beta\gamma\colon \Phi'_A$. We write $r_{gen} \overset{r_B}{\vdash} r_{spec}$.

The *rule generalization step* :

$$\frac{A(\sigma_A)\rightarrow\alpha\beta\gamma\colon \Phi'_A \qquad B(\sigma_B)\rightarrow\beta\colon \Phi_B}{A(\sigma_A)\rightarrow\alpha B(\sigma_B^*)\gamma\colon \Phi_A}$$

is *E_R-parsing-preserving*, if there exists $\sigma_A \in E_R$ and $r_{spec} \overset{*G'}{\Rightarrow} \sigma_A$ and $r_{gen} \overset{*G}{\Rightarrow} \sigma_A$. We write $r_{spec} \overset{r_B}{\dashv} r_{gen}$.

Since σ_A is a representative example, it is derived in the minimum number of derivation steps, and thus the rule r_B is always an ordered, non-recursive rule.

The goal of the rule specialization step is to obtain a new target grammar G' from G by modifying a rule of G. Similarly, the goal of the rule generalization step is to obtain a new target grammar G from G' by modifying a rule of G'. They are not to be taken as the derivation/reduction concepts in parsing. The specialization/generalization steps are the inverse of each other. From both the specialization and the generalization step we have that: $L_\sigma(r_{gen}) \supseteq L_\sigma(r_{spec})$.

In Figure 2, the specialization step $r_2 \overset{r_1}{\vdash} r_2'$ is E_R-parsing-preserving, because the rule r_2' ground-derives the syntagma *loud noise*. If instead we would have a specialization step $r_2 \overset{r_2}{\vdash} r_2''$ ($r_2'' = N1 \rightarrow Adj\ Adj\ N1$), it would not be E_R-parsing-preserving since the syntagma *loud noise* could no longer be ground-derived from the rule r_2'' (which requires two adjectives).

Definition 5. A grammar G' is *one-step specialized* from a grammar G, $G \overset{r_1}{\vdash} G'$, if $\exists r, r_1 \in P_G$ and $\exists r', r_1 \in P_{G'}$, s.t. $r \overset{r_1}{\vdash} r'$, and $\forall q \neq r, q \in P_G$ iff $q \in P_{G'}$. A grammar G' is *specialized* from a grammar G, $G \overset{*}{\vdash} G'$, if it is obtained from G in n-specialization steps: $G \overset{r_1}{\vdash} \cdots \overset{r_n}{\vdash} G'$, where n is finite. We extend the notation so that we have $G \overset{*}{\vdash} G$. Similarly, we define the concept of a grammar G *g*eneralized from a grammar G', $G' \overset{*}{\dashv} G$ using the rule generalization step.

In Figure 2, the grammar $\perp$ is one-step specialized from the grammar G_1, i.e., $G_1 \overset{r_1}{\vdash} \perp$, since $\perp$ preserve the parsing of the representative examples E_R. A grammar which contains the rule $r_2'' = N1 \rightarrow Adj\ Adj N1$ instead of r_2' is not specialized from the grammar G_1 since it does not preserve the parsing of the representative example set, E_R. Such grammars will not be in the lattice.

In order to define the grammar lattice we need to introduce one more concept: a *normalized* grammar w.r.t. a sublanguage.

Definition 6. A LWFG G is called *normalized* w.r.t. a sublanguage E_σ (not necessarily of G), if none of the grammar rules r_{spec} of G can be further generalized to a rule r_{gen} by the rule generalization step such that $\mathbb{S}(r_{spec}) \subset \mathbb{S}(r_{gen})$.

In Figure 2, grammar $\top$ is normalized w.r.t. E_σ, while $\perp$, G_1 and G_2 are not.

We now define a grammar lattice $\mathfrak{L}$ which will be the search space for our grammar learning model. We first define the set of lattice elements $\mathcal{L}$.

Let $\top$ be a LWFG, normalized and unambiguous w.r.t. a sublanguage $E_\sigma \subseteq L_\sigma(\top)$ which includes the representative example set E_R of the grammar $\top$ ($E_\sigma \supseteq E_R$). Let $\mathcal{L} = \{G | \top \overset{*}{\vdash} G\}$ be the set of grammars specialized from $\top$. We call $\top$ the *top element* of $\mathcal{L}$, and $\perp$ the *bottom element* of $\mathcal{L}$, if $\forall G \in \mathcal{L}, \top \overset{*}{\vdash} G \wedge G \overset{*}{\vdash} \perp$. The bottom element, $\perp$, is the grammar specialized from $\top$, such that the right-hand side of all grammar rules contains only preterminals. We have $\mathbb{S}(\top) = E_\sigma$ and $\mathbb{S}(\perp) \supseteq E_R$.

The grammars in $\mathcal{L}$ have the following two properties (Muresan, 2006):

- For two grammars G and G', we have that G' is specialized from G if and only if G is generalized from G', with $L_\sigma(G) \supseteq L_\sigma(G')$.
- All grammars in $\mathcal{L}$ preserve the parsing of the representative example set E_R.

Note that we have that for $G, G' \in \mathcal{L}$, if $G \overset{*}{\vdash} G'$ then $\mathbb{S}(G) \supseteq \mathbb{S}(G')$.

The system $\mathfrak{L} = \langle \mathcal{L}, \overset{*}{\vdash} \rangle$ is a complete grammar lattice (see (Muresan, 2006) for the full formal proof). In Figure 2 the grammars G_1, G_2, $\top$, $\perp$ preserve the parsing of the representative examples E_R. We have that $\top \overset{r_1}{\vdash} G_1$, $\top \overset{r_1}{\vdash} G_2$, $G_2 \overset{r_1}{\vdash} \perp$, $G_1 \overset{r_1}{\vdash} \perp$ and $\top \overset{*}{\vdash} \perp$. Due to space limitation we do not define here the *least upper bound* (lub), $\curlyvee$ and the *greatest lower bound* (glb), $\curlywedge$ operators, but in this example $\top = G_1 \curlyvee G_2$, $\perp = G_1 \curlywedge G_2$.

In oder to give a learnability theorem we need to show that $\perp$ and $\top$ elements of the lattice can be built. First, an assumption in our learning model is that the rules corresponding to the grammar preterminals are given. Thus, for a given set of representative examples, E_R, we can build the grammar $\perp$ using a bottom-up robust parser, which returns partial analyses (chunks) if it cannot return a full parse. In order to soundly build the $\top$ element of the grammar lattice from the $\perp$ grammar through generalization, we must give the definition of a grammar G *conformal* w.r.t. E_σ.

Definition 7. A LWFG G is *conformal* w.r.t. a sublanguage $E_\sigma \subseteq L_\sigma(G)$ iff G is normalized and unambiguous w.r.t. E_σ and the rule specialization step guarantees that $\mathbb{S}(r_{gen}) \supset \mathbb{S}(r_{spec})$ for all grammars specialized from G.

The only rule generalization steps allowed in the grammar induction process are those which guarantee the same relation $\mathbb{S}(r_{spec}) \subset \mathbb{S}(r_{gen})$, which ensures that all the generalized grammars belong to the grammar lattice.

In Figure 2, $\top$ is conformal to the given sublanguage E_σ. If the sublanguage were $E^*_\sigma = E_R \cup \{$*clear loud noise*$\}$ then $\top$ would not be conformal to E^*_σ since $\mathbb{S}(\top) = \mathbb{S}(G_1) = E^*_\sigma$ and thus the specialization step would not satisfy the relation $\mathbb{S}(N2 \rightarrow Det\ N1) \supset \mathbb{S}(N2 \rightarrow Det\ Noun)$. During learning, the generalization step cannot generalize from grammar G_1 to $\top$.

Theorem 1 (Learnability Theorem). *If E_R is the set of representative examples associated with a LWFG G conformal w.r.t. a sublanguage $E_\sigma \supseteq E_R$, then G can always be learned from E_R and E_σ as the grammar lattice top element ($\top = G$).*

The proof is given in (Muresan, 2006).

If the hypothesis of Theorem 1 holds, then any grammar induction algorithm that uses the complete lattice search space can converge to the lattice top element, using different search strategies. In the next section we present our new model of grammar learning which relies on the property of the search space as grammar lattice.

4 Grammar Induction Model

Based on the theoretical foundation of the hypothesis search space for LWFG learning given in the previous section, we define our grammar induction model. First, we present the LWFG induction as an Inductive Logic Programming problem. Second, we present our new relational learning model for LWFG induction, called *Grammar Approximation by Representative Sublanguage (GARS)*.

4.1 Grammar Induction Problem in ILP-setting

Inductive Logic Programming (ILP) is a class of relational learning methods concerned with inducing first-order Horn clauses from examples and background knowledge. Kietz and Džeroski (1994) have formally defined the *ILP-learning problem* as the tuple $\langle \vdash, LB, LE, LH \rangle$, where $\vdash$ is the provability relation (also called the generalization model), LB is the language of the background knowledge, LE is the language of the (positive and negative) examples, and LH is the hypothesis language. The general ILP-learning problem is undecidable. Possible choices to restrict the ILP-problem are: the provability relation, $\vdash$, the background knowledge and the hypothesis language. Research in ILP has presented positive results only for very limited subclasses of first-order logic (Kietz and Džeroski, 1994; Cohen, 1995), which are not appropriate to model natural language grammars.

Our grammar induction problem can be formulated as an ILP-learning problem $\langle \vdash, LB, LE, LH \rangle$ as follows:

- The provability relation, $\vdash$, is given by robust parsing, and we denote it by $\vdash_{rp}$. We use the "parsing as deduction" technique (Shieber et al., 1995). For all syntagmas we can say in polynomial time whether they belong or not to the grammar language. Thus, using the $\vdash_{rp}$ as generalization model, our grammar induction problem is decidable.

- The language of background knowledge, LB, is the set of LWFG rules that are already learned together with elementary syntagmas (i.e., corresponding to the lexicon), which are ground atoms (the variables are made constants).

- The language of examples, LE are syntagmas of the representative sublanguage, which are ground atoms. We only have positive examples.

- The hypothesis language, LH, is a LWFG lattice whose top element is a conformal grammar, and which preserve the parsing of representative examples.

4.2 Grammar Approximation by Representative Sublanguage Model

We have formulated the grammar induction problem in the ILP-setting. The theoretical learning model,

called *Grammar Approximation by Representative Sublanguage (GARS)*, can be formulated as follows:
Given:

- a representative example set E_R, lexically consistent (i.e., it allows the construction of the grammar lattice $\perp$ element)
- a finite sublanguage E_σ, conformal and thus unambiguous, which includes the representative example set, $E_\sigma \supseteq E_R$. We called this sublanguage, the *representative sublanguage*

Learn a grammar G, using the above ILP-learning setting, such that G is unique and $E_\sigma \subseteq L_\sigma(G)$.

The hypothesis space is a complete grammar lattice, and thus the uniqueness property of the learned grammar is guaranteed by the learnability theorem (i.e., the learned grammar is the lattice top element). This learnability result extends significantly the class of problems learnable by ILP methods.

The GARS model uses two polynomial algorithms for LWFG learning. In the first algorithm, the learner is presented with an ordered set of representative examples (syntagmas), i.e., the examples are ordered from the simplest to the most complex. The reader should remember that for a LWFG G, there exists a partial ordering among the grammar nonterminals, which allows a total ordering of the representative examples of the grammar G. Thus, in this algorithm, the learner has access to the ordered representative syntagmas when learning the grammar. However, in practice it might be difficult to provide the learner with the "true" order of examples, especially when modeling complex language phenomena. The second algorithm is an iterative algorithm that learns starting from a random order of the representative example set. Due to the property of the search space, both algorithms converge to the same target grammar.

Using ILP and theory revision terminology (Greiner, 1999), we can establish the following analogy: syntagmas (examples) are "labeled queries", the LWFG lattice is the "space of theories", and a LWFG in the lattice is "a theory." The first algorithm learns from an "empty theory", while the second algorithm is an instance of "theory revision", since the grammar ("theory") learned during the first iteration, is then revised, by deleting and adding rules.

Both of these algorithms are cover set algorithms. In the first step the most specific grammar rule is generated from the current representative example. The category name annotated in the representative example gives the name of the lhs nonterminal (predicate invention in ILP terminology), while the robust parser returns the minimum number of chunks that cover the representative example. In the second step this most specific rule is generalized using as performance criterion the number of the examples in E_σ that can be parsed using the candidate grammar rule (hypothesis) together with the previous learned rules. For the full details for these two algorithms, and the proof of their polynomial efficiency, we refer the reader to (Muresan, 2006).

5 Discussion

A practical advantage of our GARS model is that instead of writing syntactic-semantic grammars by hand (both rules and constraints), we construct just a small annotated treebank - utterances and their semantic molecules. If the grammar needs to be refined, or enhanced, we only refine, or enhance the representative examples/sublanguage, and not the grammar rules and constraints, which would be a more difficult task.

We have built a framework to test whether our GARS model can learn diverse and complex linguistic phenomena. We have primarily analyzed a set of definitional-type sentences in the medical domain. The phenomena covered by our learned grammar includes complex noun phrases (including noun compounds, nominalization), prepositional phrases, relative clauses and reduced relative clauses, finite and non-finite verbal constructions (including, tense, aspect, negation, and subject-verb agreement), copula *to be*, and raising and control constructions. We also learned rules for wh-questions (including long-distance dependencies). In Figure 3 we show the ontology-level representation of a definition-type sentence obtained using our learned grammar. It includes the treatment of reduced relative clauses, raising construction (*tends to persist*, where *virus* is not the argument of *tends* but the argument of *persist*), and noun compounds. The learned grammar together with a semantic interpreter targeted to terminological knowledge has been used in an acquisition-query experiment, where the answers are at the concept level (the querying is a graph

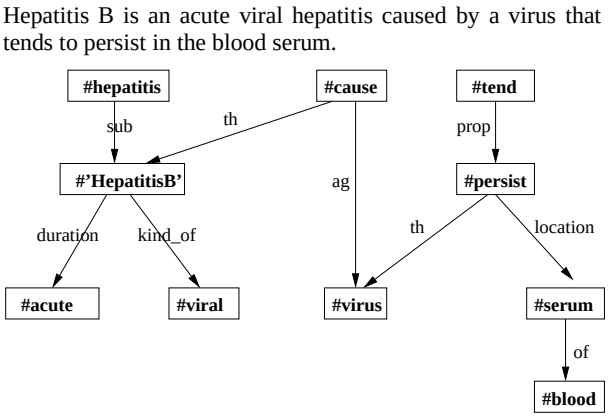


Figure 3: A definition-type sentence and its ontology-based representation obtained using our learned LWFG

matching problem where the "wh-word" matches the answer concept). A detailed discussion of the linguistic phenomena covered by our learned grammar using the GARS model, as well as the use of this grammar for terminological knowledge acquisition, is given in (Muresan, 2006).

To learn the grammar used in these experiments we annotated 151 representative examples and 448 examples used as a representative sublanguage for generalization. Annotating these examples requires knowledge about categories and their attributes. We used 31 categories (nonterminals) and 37 attributes (e.g., category, head, number, person). In this experiment, we chose the representative examples guided by the type of phenomena we wanted to modeled and which occurred in our corpus. We also used 13 lexical categories (i.e., parts of speech). The learned grammar contains 151 rules and 151 constraints.

6 Conclusion

We have presented Lexicalized Well-Founded Grammars, a type of constraint-based grammars for natural language specifically designed to enable learning from representative examples annotated with semantics. We have presented a new grammar learning model and showed that the search space is a complete grammar lattice that guarantees the uniqueness of the learned grammar. Starting from these fundamental theoretical results, there are several directions into which to take this research. A first obvious extension is to have probabilistic-LWFGs. For example, the ontology constraints might not be "hard" constraints, but "soft" ones (because language expressions are more or less likely to be used in a certain context). Investigating where to add probabilities (ontology, grammar rules, or both) is part of our planned future work. Another future extension of this work is to investigate how to automatically select the representative examples from an existing treebank.

References

Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of COLING-04*.

William Cohen. 1995. Pac-learning recursive logic programs: Negative results. *Journal of Artificial Intelligence Research*, 2:541–573.

Rusins Freivalds, Efim B. Kinber, and Rolf Wiehagen. 1993. On the power of inductive inference from good examples. *Theoretical Computer Science*, 110(1):131–144.

R. Ge and R.J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of CoNLL-2005*.

Russell Greiner. 1999. The complexity of theory revision. *Artificial Intelligence Journal*, 107(2):175–217.

Aria Haghighi and Dan Klein. 2006. Prototype-driven grammar induction. In *Proceedings of ACL'06*.

Jörg-Uwe Kietz and Sašo Džeroski. 1994. Inductive logic programming and learnability. *ACM SIGART Bulletin.*, 5(1):22–32.

Smaranda Muresan. 2006. *Learning Constraint-based Grammars from Representative Examples: Theory and Applications*. Ph.D. thesis, Columbia University. http://www1.cs.columbia.edu/∼smara/muresan_thesis.pdf.

Fernando C. Pereira and David H.D Warren. 1980. Definite Clause Grammars for language analysis. *Artificial Intelligence*, 13:231–278.

Stuart Shieber, Hans Uszkoreit, Fernando Pereira, Jane Robinson, and Mabry Tyson. 1983. The formalism and implementation of PATR-II. In Barbara J. Grosz and Mark Stickel, editors, *Research on Interactive Acquisition and Use of Knowledge*, pages 39–79. SRI International, Menlo Park, CA, November.

Stuart Shieber, Yves Schabes, and Fernando Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1-2):3–36.

Shuly Wintner. 1999. Compositional semantics for linguistic formalisms. In *Proceedings of the ACL'99*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI-05*.

Chinese Segmentation with a Word-Based Perceptron Algorithm

Yue Zhang and **Stephen Clark**
Oxford University Computing Laboratory
Wolfson Building, Parks Road
Oxford OX1 3QD, UK
{yue.zhang,stephen.clark}@comlab.ox.ac.uk

Abstract

Standard approaches to Chinese word segmentation treat the problem as a tagging task, assigning labels to the characters in the sequence indicating whether the character marks a word boundary. Discriminatively trained models based on local character features are used to make the tagging decisions, with Viterbi decoding finding the highest scoring segmentation. In this paper we propose an alternative, word-based segmentor, which uses features based on complete words and word sequences. The generalized perceptron algorithm is used for discriminative training, and we use a beam-search decoder. Closed tests on the first and second SIGHAN bakeoffs show that our system is competitive with the best in the literature, achieving the highest reported F-scores for a number of corpora.

1 Introduction

Words are the basic units to process for most NLP tasks. The problem of Chinese word segmentation (CWS) is to find these basic units for a given sentence, which is written as a continuous sequence of characters. It is the initial step for most Chinese processing applications.

Chinese character sequences are ambiguous, often requiring knowledge from a variety of sources for disambiguation. Out-of-vocabulary (OOV) words are a major source of ambiguity. For example, a difficult case occurs when an OOV word consists of characters which have themselves been seen as words; here an automatic segmentor may split the OOV word into individual single-character words. Typical examples of unseen words include Chinese names, translated foreign names and idioms.

The segmentation of known words can also be ambiguous. For example, "这里面" should be "这 里 (here) 面 (flour)" in the sentence "这里面和米很贵" (flour and rice are expensive here) or "这 (here) 里面 (inside)" in the sentence "这里面很冷" (it's cold inside here). The ambiguity can be resolved with information about the neighboring words. In comparison, for the sentences "洽谈会很成功", possible segmentations include "洽谈 (the discussion) 会 (will) 很 (very) 成功 (be successful)" and "洽谈会 (the discussion meeting) 很 (very) 成功 (be successful)". The ambiguity can only be resolved with contextual information outside the sentence. Human readers often use semantics, contextual information about the document and world knowledge to resolve segmentation ambiguities.

There is no fixed standard for Chinese word segmentation. Experiments have shown that there is only about 75% agreement among native speakers regarding the correct word segmentation (Sproat et al., 1996). Also, specific NLP tasks may require different segmentation criteria. For example, "北京银行" could be treated as a single word (Bank of Beijing) for machine translation, while it is more naturally segmented into "北京 (Beijing) 银行 (bank)" for tasks such as text-to-speech synthesis. Therefore, supervised learning with specifically defined training data has become the dominant approach.

Following Xue (2003), the standard approach to

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 840–847,
Prague, Czech Republic, June 2007.

supervised learning for CWS is to treat it as a tagging task. Tags are assigned to each character in the sentence, indicating whether the character is a single-character word or the start, middle or end of a multi-character word. The features are usually confined to a five-character window with the current character in the middle. In this way, dynamic programming algorithms such as the Viterbi algorithm can be used for decoding.

Several discriminatively trained models have recently been applied to the CWS problem. Examples include Xue (2003), Peng et al. (2004) and Shi and Wang (2007); these use maximum entropy (ME) and conditional random field (CRF) models (Ratnaparkhi, 1998; Lafferty et al., 2001). An advantage of these models is their flexibility in allowing knowledge from various sources to be encoded as features.

Contextual information plays an important role in word segmentation decisions; especially useful is information about surrounding words. Consider the sentence "中国外企业", which can be from "其中 (among which) 国外 (foreign) 企业 (companies)", or "中国 (in China) 外企 (foreign companies) 业务 (business)". Note that the five-character window surrounding "外" is the same in both cases, making the tagging decision for that character difficult given the local window. However, the correct decision can be made by comparison of the two three-word windows containing this character.

In order to explore the potential of word-based models, we adapt the perceptron discriminative learning algorithm to the CWS problem. Collins (2002) proposed the perceptron as an alternative to the CRF method for HMM-style taggers. However, our model does not map the segmentation problem to a tag sequence learning problem, but defines features on segmented sentences directly. Hence we use a beam-search decoder during training and testing; our idea is similar to that of Collins and Roark (2004) who used a beam-search decoder as part of a perceptron parsing model. Our work can also be seen as part of the recent move towards *search-based* learning methods which do not rely on dynamic programming and are thus able to exploit larger parts of the context for making decisions (Daume III, 2006).

We study several factors that influence the performance of the perceptron word segmentor, including the averaged perceptron method, the size of the beam and the importance of word-based features. We compare the accuracy of our final system to the state-of-the-art CWS systems in the literature using the first and second SIGHAN bakeoff data. Our system is competitive with the best systems, obtaining the highest reported F-scores on a number of the bakeoff corpora. These results demonstrate the importance of word-based features for CWS. Furthermore, our approach provides an example of the potential of search-based discriminative training methods for NLP tasks.

2 The Perceptron Training Algorithm

We formulate the CWS problem as finding a mapping from an input sentence $x \in X$ to an output sentence $y \in Y$, where X is the set of possible raw sentences and Y is the set of possible segmented sentences. Given an input sentence x, the correct output segmentation $F(x)$ satisfies:

$$F(x) = \arg\max_{y \in \text{GEN}(x)} \text{Score}(y)$$

where $\text{GEN}(x)$ denotes the set of possible segmentations for an input sentence x, consistent with notation from Collins (2002).

The score for a segmented sentence is computed by first mapping it into a set of features. A feature is an indicator of the occurrence of a certain pattern in a segmented sentence. For example, it can be the occurrence of "里面" as a single word, or the occurrence of "里" separated from "面" in two adjacent words. By defining features, a segmented sentence is mapped into a global feature vector, in which each dimension represents the count of a particular feature in the sentence. The term "global" feature vector is used by Collins (2002) to distinguish between feature count vectors for whole sequences and the "local" feature vectors in ME tagging models, which are Boolean valued vectors containing the indicator features for one element in the sequence.

Denote the global feature vector for segmented sentence y with $\Phi(y) \in R^d$, where d is the total number of features in the model; then $\text{Score}(y)$ is computed by the dot product of vector $\Phi(y)$ and a parameter vector $\overline{\alpha} \in R^d$, where α_i is the weight for the ith feature:

$$\text{Score}(y) = \Phi(y) \cdot \overline{\alpha}$$

Inputs: training examples (x_i, y_i)
Initialization: set $\overline{\alpha} = 0$
Algorithm:
  for $t = 1..T$, $i = 1..N$
    calculate $z_i = \arg\max_{y \in \text{GEN}(x_i)} \Phi(y) \cdot \overline{\alpha}$
    if $z_i \neq y_i$
      $\overline{\alpha} = \overline{\alpha} + \Phi(y_i) - \Phi(z_i)$
Outputs: $\overline{\alpha}$

Figure 1: the perceptron learning algorithm, adapted from Collins (2002)

The perceptron training algorithm is used to determine the weight values $\overline{\alpha}$.

The training algorithm initializes the parameter vector as all zeros, and updates the vector by decoding the training examples. Each training sentence is turned into the raw input form, and then decoded with the current parameter vector. The output segmented sentence is compared with the original training example. If the output is incorrect, the parameter vector is updated by adding the global feature vector of the training example and subtracting the global feature vector of the decoder output. The algorithm can perform multiple passes over the same training sentences. Figure 1 gives the algorithm, where N is the number of training sentences and T is the number of passes over the data.

Note that the algorithm from Collins (2002) was designed for discriminatively training an HMM-style tagger. Features are extracted from an input sequence x and its corresponding tag sequence y:

$$\text{Score}(x, y) = \Phi(x, y) \cdot \overline{\alpha}$$

Our algorithm is not based on an HMM. For a given input sequence x, even the length of different candidates y (the number of words) is not fixed. Because the output sequence y (the segmented sentence) contains all the information from the input sequence x (the raw sentence), the global feature vector $\Phi(x, y)$ is replaced with $\Phi(y)$, which is extracted from the candidate segmented sentences directly.

Despite the above differences, since the theorems of convergence and their proof (Collins, 2002) are only dependent on the feature vectors, and not on the source of the feature definitions, the perceptron algorithm is applicable to the training of our CWS model.

2.1 The averaged perceptron

The averaged perceptron algorithm (Collins, 2002) was proposed as a way of reducing overfitting on the training data. It was motivated by the voted-perceptron algorithm (Freund and Schapire, 1999) and has been shown to give improved accuracy over the non-averaged perceptron on a number of tasks. Let N be the number of training sentences, T the number of training iterations, and $\overline{\alpha}^{n,t}$ the parameter vector immediately after the nth sentence in the tth iteration. The averaged parameter vector $\overline{\gamma} \in R^d$ is defined as:

$$\overline{\gamma} = \frac{1}{NT} \sum_{n=1..N, t=1..T} \overline{\alpha}^{n,t}$$

To compute the averaged parameters $\overline{\gamma}$, the training algorithm in Figure 1 can be modified by keeping a total parameter vector $\overline{\sigma}^{n,t} = \sum \overline{\alpha}^{n,t}$, which is updated using $\overline{\alpha}$ after each training example. After the final iteration, $\overline{\gamma}$ is computed as $\overline{\sigma}^{n,t}/NT$. In the averaged perceptron algorithm, $\overline{\gamma}$ is used instead of $\overline{\alpha}$ as the final parameter vector.

With a large number of features, calculating the total parameter vector $\overline{\sigma}^{n,t}$ after each training example is expensive. Since the number of changed dimensions in the parameter vector $\overline{\alpha}$ after each training example is a small proportion of the total vector, we use a lazy update optimization for the training process.[1] Define an update vector $\overline{\tau}$ to record the number of the training sentence n and iteration t when each dimension of the averaged parameter vector was last updated. Then after each training sentence is processed, only update the dimensions of the total parameter vector corresponding to the features in the sentence. (Except for the last example in the last iteration, when each dimension of $\overline{\tau}$ is updated, no matter whether the decoder output is correct or not).

Denote the sth dimension in each vector before processing the nth example in the tth iteration as $\alpha_s^{n-1,t}$, $\sigma_s^{n-1,t}$ and $\tau_s^{n-1,t} = (n_{\tau,s}, t_{\tau,s})$. Suppose that the decoder output $z_{n,t}$ is different from the training example y_n. Now $\alpha_s^{n,t}$, $\sigma_s^{n,t}$ and $\tau_s^{n,t}$ can

[1] Daume III (2006) describes a similar algorithm.

be updated in the following way:

$$\sigma_s^{n,t} = \sigma_s^{n-1,t} + \alpha_s^{n-1,t} \times (tN + n - t_{\tau,s}N - n_{\tau,s})$$
$$\alpha_s^{n,t} = \alpha_s^{n-1,t} + \Phi(y_n) - \Phi(z_{n,t})$$
$$\sigma_s^{n,t} = \sigma_s^{n,t} + \Phi(y_n) - \Phi(z_{n,t})$$
$$\tau_s^{n,t} = (n, t)$$

We found that this lazy update method was significantly faster than the naive method.

3 The Beam-Search Decoder

The decoder reads characters from the input sentence one at a time, and generates candidate segmentations incrementally. At each stage, the next incoming character is combined with an existing candidate in two different ways to generate new candidates: it is either appended to the last word in the candidate, or taken as the start of a new word. This method guarantees exhaustive generation of possible segmentations for any input sentence.

Two agendas are used: the source agenda and the target agenda. Initially the source agenda contains an empty sentence and the target agenda is empty. At each processing stage, the decoder reads in a character from the input sentence, combines it with each candidate in the source agenda and puts the generated candidates onto the target agenda. After each character is processed, the items in the target agenda are copied to the source agenda, and then the target agenda is cleaned, so that the newly generated candidates can be combined with the next incoming character to generate new candidates. After the last character is processed, the decoder returns the candidate with the best score in the source agenda. Figure 2 gives the decoding algorithm.

For a sentence with length l, there are 2^{l-1} different possible segmentations. To guarantee reasonable running speed, the size of the target agenda is limited, keeping only the B best candidates.

4 Feature templates

The feature templates are shown in Table 1. Features 1 and 2 contain only word information, 3 to 5 contain character and length information, 6 and 7 contain only character information, 8 to 12 contain word and character information, while 13 and 14 contain

Input: raw sentence $sent$ – a list of characters
Initialization: set agendas $src = [[]]$, $tgt = []$
Variables: candidate sentence $item$ – a list of words
Algorithm:
  for $index = 0..sent$.length-1:
    var $char = sent[index]$
    foreach $item$ in src:
      // append as a new word to the candidate
      var $item_1 = item$
      $item_1$.append($char$.toWord())
      tgt.insert($item_1$)
      // append the character to the last word
      if $item$.length > 1:
        var $item_2 = item$
        $item_2[item_2$.length$-1]$.append($char$)
        tgt.insert($item_2$)
    $src = tgt$
    $tgt = []$
Outputs: src.best_item

Figure 2: The decoding algorithm

word and length information. Any segmented sentence is mapped to a global feature vector according to these templates. There are $356,337$ features with non-zero values after 6 training iterations using the development data.

For this particular feature set, the longest range features are word bigrams. Therefore, among partial candidates ending with the same bigram, the best one will also be in the best final candidate. The decoder can be optimized accordingly: when an incoming character is combined with candidate items as a new word, only the best candidate is kept among those having the same last word.

5 Comparison with Previous Work

Among the character-tagging CWS models, Li et al. (2005) uses an uneven margin alteration of the traditional perceptron classifier (Li et al., 2002). Each character is classified independently, using information in the neighboring five-character window. Liang (2005) uses the discriminative perceptron algorithm (Collins, 2002) to score whole character tag sequences, finding the best candidate by the global score. It can be seen as an alternative to the ME and CRF models (Xue, 2003; Peng et al., 2004), which

1	word w
2	word bigram w_1w_2
3	single-character word w
4	a word starting with character c and having length l
5	a word ending with character c and having length l
6	space-separated characters c_1 and c_2
7	character bigram c_1c_2 in any word
8	the first and last characters c_1 and c_2 of any word
9	word w immediately before character c
10	character c immediately before word w
11	the starting characters c_1 and c_2 of two consecutive words
12	the ending characters c_1 and c_2 of two consecutive words
13	a word of length l and the previous word w
14	a word of length l and the next word w

Table 1: feature templates

do not involve word information. Wang et al. (2006) incorporates an N-gram language model in ME tagging, making use of word information to improve the character tagging model. The key difference between our model and the above models is the word-based nature of our system.

One existing method that is based on sub-word information, Zhang et al. (2006), combines a CRF and a rule-based model. Unlike the character-tagging models, the CRF submodel assigns tags to sub-words, which include single-character words and the most frequent multiple-character words from the training corpus. Thus it can be seen as a step towards a word-based model. However, sub-words do not necessarily contain full word information. Moreover, sub-word extraction is performed separately from feature extraction. Another difference from our model is the rule-based submodel, which uses a dictionary-based forward maximum match method described by Sproat et al. (1996).

6 Experiments

Two sets of experiments were conducted. The first, used for development, was based on the part of Chinese Treebank 4 that is not in Chinese Treebank 3 (since CTB3 was used as part of the first bakeoff). This corpus contains 240K characters (150K words and 4798 sentences). 80% of the sentences (3813) were randomly chosen for training and the rest (985 sentences) were used as development testing data. The accuracies and learning curves for the non-averaged and averaged perceptron were compared. The influence of particular features and the agenda size were also studied.

The second set of experiments used training and testing sets from the first and second international Chinese word segmentation bakeoffs (Sproat and Emerson, 2003; Emerson, 2005). The accuracies are compared to other models in the literature.

F-measure is used as the accuracy measure. Define precision p as the percentage of words in the decoder output that are segmented correctly, and recall r as the percentage of gold standard output words that are correctly segmented by the decoder. The (balanced) F-measure is $2pr/(p+r)$.

CWS systems are evaluated by two types of tests. The *closed* tests require that the system is trained only with a designated training corpus. Any extra knowledge is not allowed, including common surnames, Chinese and Arabic numbers, European letters, lexicons, part-of-speech, semantics and so on. The *open* tests do not impose such restrictions.

Open tests measure a model's capability to utilize extra information and domain knowledge, which can lead to improved performance, but since this extra information is not standardized, direct comparison between open test results is less informative.

In this paper, we focus only on the closed test. However, the perceptron model allows a wide range of features, and so future work will consider how to integrate open resources into our system.

6.1 Learning curve

In this experiment, the agenda size was set to 16, for both training and testing. Table 2 shows the precision, recall and F-measure for the development set after 1 to 10 training iterations, as well as the number of mistakes made in each iteration. The corresponding learning curves for both the non-averaged and averaged perceptron are given in Figure 3.

The table shows that the number of mistakes made in each iteration decreases, reflecting the convergence of the learning algorithm. The averaged per-

Iteration	1	2	3	4	5	6	7	8	9	10
P (non-avg)	89.0	91.6	92.0	92.3	92.5	92.5	92.5	92.7	92.6	92.6
R (non-avg)	88.3	91.4	92.2	92.6	92.7	92.8	93.0	93.0	93.1	93.2
F (non-avg)	88.6	91.5	92.1	92.5	92.6	92.6	92.7	92.8	92.8	92.9
P (avg)	91.7	92.8	93.1	92.2	93.1	93.2	93.2	93.2	93.2	93.2
R (avg)	91.6	92.9	93.3	93.4	93.4	93.5	93.5	93.5	93.6	93.6
F (avg)	91.6	92.9	93.2	93.3	93.3	93.4	93.3	93.3	93.4	93.4
#Wrong sentences	3401	1652	945	621	463	288	217	176	151	139

Table 2: accuracy using non-averaged and averaged perceptron.
P - precision (%), R - recall (%), F - F-measure.

B	2	4	8	16	32	64	128	256	512	1024
Tr	660	610	683	830	1111	1645	2545	4922	9104	15598
Seg	18.65	18.18	28.85	26.52	36.58	56.45	95.45	173.38	325.99	559.87
F	86.90	92.95	93.33	93.38	93.25	93.29	93.19	93.07	93.24	93.34

Table 3: the influence of agenda size.
B - agenda size, Tr - training time (seconds), Seg - testing time (seconds), F - F-measure.

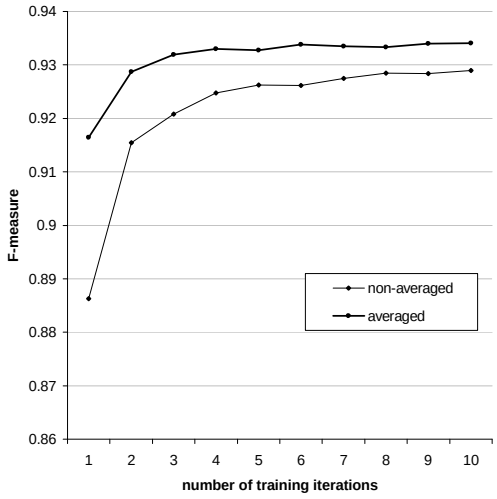


Figure 3: learning curves of the averaged and non-averaged perceptron algorithms

ceptron algorithm improves the segmentation accuracy at each iteration, compared with the non-averaged perceptron. The learning curve was used to fix the number of training iterations at 6 for the remaining experiments.

6.2 The influence of agenda size

Reducing the agenda size increases the decoding speed, but it could cause loss of accuracy by eliminating potentially good candidates. The agenda size also affects the training time, and resulting model, since the perceptron training algorithm uses the decoder output to adjust the model parameters. Table 3 shows the accuracies with ten different agenda sizes, each used for both training and testing.

Accuracy does not increase beyond $B = 16$. Moreover, the accuracy is quite competitive even with B as low as 4. This reflects the fact that the best segmentation is often within the current top few candidates in the agenda.[2] Since the training and testing time generally increases as N increases, the agenda size is fixed to 16 for the remaining experiments.

6.3 The influence of particular features

Our CWS model is highly dependent upon word information. Most of the features in Table 1 are related to words. Table 4 shows the accuracy with various features from the model removed.

Among the features, vocabulary words (feature 1) and length prediction by characters (features 3 to 5) showed strong influence on the accuracy, while word bigrams (feature 2) and special characters in them (features 11 and 12) showed comparatively weak influence.

[2]The optimization in Section 4, which has a pruning effect, was applied to this experiment. Similar observations were made in separate experiments without such optimization.

Features	F	Features	F
All	93.38	w/o 1	92.88
w/o 2	93.36	w/o 3, 4, 5	92.72
w/o 6	93.13	w/o 7	93.13
w/o 8	93.14	w/o 9, 10	93.31
w/o 11, 12	93.38	w/o 13, 14	93.23

Table 4: the influence of features. (F: F-measure. Feature numbers are from Table 1)

6.4 Closed test on the SIGHAN bakeoffs

Four training and testing corpora were used in the first bakeoff (Sproat and Emerson, 2003), including the Academia Sinica Corpus (AS), the Penn Chinese Treebank Corpus (CTB), the Hong Kong City University Corpus (CU) and the Peking University Corpus (PU). However, because the testing data from the Penn Chinese Treebank Corpus is currently unavailable, we excluded this corpus. The corpora are encoded in GB (PU, CTB) and BIG5 (AS, CU). In order to test them consistently in our system, they are all converted to UTF8 without loss of information.

The results are shown in Table 5. We follow the format from Peng et al. (2004). Each row represents a CWS model. The first eight rows represent models from Sproat and Emerson (2003) that participated in at least one closed test from the table, row "Peng" represents the CRF model from Peng et al. (2004), and the last row represents our model. The first three columns represent tests with the AS, CU and PU corpora, respectively. The best score in each column is shown in bold. The last two columns represent the average accuracy of each model over the tests it participated in (SAV), and our average over the same tests (OAV), respectively. For each row the best average is shown in bold.

We achieved the best accuracy in two of the three corpora, and better overall accuracy than the majority of the other models. The average score of S10 is 0.7% higher than our model, but S10 only participated in the HK test.

Four training and testing corpora were used in the second bakeoff (Emerson, 2005), including the Academia Sinica corpus (AS), the Hong Kong City University Corpus (CU), the Peking University Corpus (PK) and the Microsoft Research Corpus (MR).

	AS	CU	PU	SAV	OAV
S01	93.8	90.1	**95.1**	93.0	**95.0**
S04			93.9	93.9	**94.0**
S05	94.2		89.4	91.8	**95.3**
S06	94.5	92.4	92.4	93.1	**95.0**
S08		90.4	93.6	92.0	**94.3**
S09	96.1		94.6	**95.4**	95.3
S10			94.7	**94.7**	94.0
S12	95.9	91.6		93.8	**95.6**
Peng	95.6	92.8	94.1	94.2	**95.0**
	96.5	**94.6**	94.0		

Table 5: the accuracies over the first SIGHAN bakeoff data.

	AS	CU	PK	MR	SAV	OAV
S14	94.7	94.3	95.0	96.4	95.1	**95.4**
S15b	**95.2**	94.1	94.1	95.8	94.8	**95.4**
S27	94.5	94.0	95.0	96.0	94.9	**95.4**
Zh-a	94.7	94.6	94.5	96.4	95.1	**95.4**
Zh-b	95.1	**95.1**	**95.1**	97.1	**95.6**	95.4
	94.6	**95.1**	94.5	**97.2**		

Table 6: the accuracies over the second SIGHAN bakeoff data.

Different encodings were provided, and the UTF8 data for all four corpora were used in this experiment.

Following the format of Table 5, the results for this bakeoff are shown in Table 6. We chose the three models that achieved at least one best score in the closed tests from Emerson (2005), as well as the sub-word-based model of Zhang et al. (2006) for comparison. Row "Zh-a" and "Zh-b" represent the pure sub-word CRF model and the confidence-based combination of the CRF and rule-based models, respectively.

Again, our model achieved better overall accuracy than the majority of the other models. One system to achieve comparable accuracy with our system is Zh-b, which improves upon the sub-word CRF model (Zh-a) by combining it with an independent dictionary-based submodel and improving the accuracy of known words. In comparison, our system is based on a single perceptron model.

In summary, closed tests for both the first and the second bakeoff showed competitive results for our

system compared with the best results in the literature. Our word-based system achieved the best F-measures over the AS (96.5%) and CU (94.6%) corpora in the first bakeoff, and the CU (95.1%) and MR (97.2%) corpora in the second bakeoff.

7 Conclusions and Future Work

We proposed a word-based CWS model using the discriminative perceptron learning algorithm. This model is an alternative to the existing character-based tagging models, and allows word information to be used as features. One attractive feature of the perceptron training algorithm is its simplicity, consisting of only a decoder and a trivial update process. We use a beam-search decoder, which places our work in the context of recent proposals for search-based discriminative learning algorithms. Closed tests using the first and second SIGHAN CWS bakeoff data demonstrated our system to be competitive with the best in the literature.

Open features, such as knowledge of numbers and European letters, and relationships from semantic networks (Shi and Wang, 2007), have been reported to improve accuracy. Therefore, given the flexibility of the feature-based perceptron model, an obvious next step is the study of open features in the segmentor.

Also, we wish to explore the possibility of incorporating POS tagging and parsing features into the discriminative model, leading to joint decoding. The advantage is two-fold: higher level syntactic information can be used in word segmentation, while joint decoding helps to prevent bottom-up error propagation among the different processing steps.

Acknowledgements

This work is supported by the ORS and Clarendon Fund. We thank the anonymous reviewers for their insightful comments.

References

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL'04*, pages 111–118, Barcelona, Spain, July.

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

Hal Daume III. 2006. *Practical Structured Learning for Natural Language Processing*. Ph.D. thesis, USC.

Thomas Emerson. 2005. The second international Chinese word segmentation bakeoff. In *Proceedings of The Fourth SIGHAN Workshop*, Jeju, Korea.

Y. Freund and R. Schapire. 1999. Large margin classification using the perceptron algorithm. In *Machine Learning*, pages 277–296.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th ICML*, pages 282–289, Massachusetts, USA.

Y. Li, Zaragoza, R. H., Herbrich, J. Shawe-Taylor, and J. Kandola. 2002. The perceptron algorithm with uneven margins. In *Proceedings of the 9th ICML*, pages 379–386, Sydney, Australia.

Yaoyong Li, Chuanjiang Miao, Kalina Bontcheva, and Hamish Cunningham. 2005. Perceptron learning for Chinese word segmentation. In *Proceedings of the Fourth SIGHAN Workshop*, Jeju, Korea.

Percy Liang. 2005. Semi-supervised learning for natural language. Master's thesis, MIT.

F. Peng, F. Feng, , and A. McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of COLING*, Geneva, Switzerland.

Adwait Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, UPenn.

Yanxin Shi and Mengqiu Wang. 2007. A dual-layer CRF based joint decoding method for cascade segmentation and labelling tasks. In *Proceedings of IJCAI*, Hyderabad, India.

Richard Sproat and Thomas Emerson. 2003. The first international Chinese word segmentation bakeoff. In *Proceedings of The Second SIGHAN Workshop*, pages 282–289, Sapporo, Japan, July.

R. Sproat, C. Shih, W. Gail, and N. Chang. 1996. A stochastic finite-state word-segmentation algorithm for Chinese. In *Computational Linguistics*, volume 22(3), pages 377–404.

Xinhao Wang, Xiaojun Lin, Dianhai Yu, Hao Tian, and Xihong Wu. 2006. Chinese word segmentation with maximum entropy and n-gram language model. In *Proceedings of the Fifth SIGHAN Workshop*, pages 138–141, Sydney, Australia, July.

N. Xue. 2003. Chinese word segmentation as character tagging. In *International Journal of Computational Linguistics and Chinese Language Processing*, volume 8(1).

Ruiqiang Zhang, Genichiro Kikui, and Eiichiro Sumita. 2006. Subword-based tagging by conditional random fields for Chinese word segmentation. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion*, volume Short Papers, pages 193–196, New York City, USA, June.

Unsupervised Coreference Resolution in a Nonparametric Bayesian Model

Aria Haghighi and Dan Klein
Computer Science Division
UC Berkeley
{aria42, klein}@cs.berkeley.edu

Abstract

We present an unsupervised, nonparametric Bayesian approach to coreference resolution which models both global entity identity across a corpus as well as the sequential anaphoric structure within each document. While most existing coreference work is driven by pairwise decisions, our model is fully generative, producing each mention from a combination of global entity properties and local attentional state. Despite being unsupervised, our system achieves a 70.3 MUC F_1 measure on the MUC-6 test set, broadly in the range of some recent supervised results.

1 Introduction

Referring to an entity in natural language can broadly be decomposed into two processes. First, speakers directly introduce new entities into discourse, entities which may be shared across discourses. This initial reference is typically accomplished with proper or nominal expressions. Second, speakers refer back to entities already introduced. This anaphoric reference is canonically, though of course not always, accomplished with pronouns, and is governed by linguistic and cognitive constraints. In this paper, we present a nonparametric generative model of a document corpus which naturally connects these two processes.

Most recent coreference resolution work has focused on the task of deciding which *mentions* (noun phrases) in a document are coreferent. The dominant approach is to decompose the task into a collection of pairwise coreference decisions. One then applies discriminative learning methods to pairs of mentions, using features which encode properties such as distance, syntactic environment, and so on (Soon et al., 2001; Ng and Cardie, 2002). Although such approaches have been successful, they have several liabilities. First, rich features require plentiful labeled data, which we do not have for coreference tasks in most domains and languages. Second, coreference is inherently a clustering or partitioning task. Naive pairwise methods can and do fail to produce coherent partitions. One classic solution is to make greedy left-to-right linkage decisions. Recent work has addressed this issue in more global ways. McCallum and Wellner (2004) use graph partioning in order to reconcile pairwise scores into a final coherent clustering. Nonetheless, all these systems crucially rely on pairwise models because cluster-level models are much harder to work with, combinatorially, in discriminative approaches.

Another thread of coreference work has focused on the problem of identifying matches between documents (Milch et al., 2005; Bhattacharya and Getoor, 2006; Daume and Marcu, 2005). These methods ignore the sequential anaphoric structure inside documents, but construct models of how and when entities are shared between them.[1] These models, as ours, are generative ones, since the focus is on cluster discovery and the data is generally unlabeled.

In this paper, we present a novel, fully generative, nonparametric Bayesian model of mentions in a document corpus. Our model captures both within- and cross-document coreference. At the top, a hierarchical Dirichlet process (Teh et al., 2006) cap-

[1] Milch et al. (2005) works with citations rather than discourses and does model the linear structure of the citations.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 848–855,
Prague, Czech Republic, June 2007.

tures cross-document entity (and parameter) sharing, while, at the bottom, a sequential model of salience captures within-document sequential structure. As a joint model of several kinds of discourse variables, it can be used to make predictions about either kind of coreference, though we focus experimentally on within-document measures. To the best of our ability to compare, our model achieves the best unsupervised coreference performance.

2 Experimental Setup

We adopt the terminology of the Automatic Context Extraction (ACE) task (NIST, 2004). For this paper, we assume that each document in a corpus consists of a set of *mentions*, typically noun phrases. Each mention is a *reference* to some entity in the domain of discourse. The coreference resolution task is to partition the mentions according to referent. Mentions can be divided into three categories, *proper* mentions (names), *nominal* mentions (descriptions), and *pronominal* mentions (pronouns).

In section 3, we present a sequence of increasingly enriched models, motivating each from shortcomings of the previous. As we go, we will indicate the performance of each model on data from ACE 2004 (NIST, 2004). In particular, we used as our development corpus the English translations of the Arabic and Chinese treebanks, comprising 95 documents and about 3,905 mentions. This data was used heavily for model design and hyperparameter selection. In section 5, we present final results for new test data from MUC-6 on which no tuning or development was performed. This test data will form our basis for comparison to previous work.

In all experiments, as is common, we will assume that we have been given as part of our input the true mention boundaries, the head word of each mention and the mention type (proper, nominal, or pronominal). For the ACE data sets, the head and mention type are given as part of the mention annotation. For the MUC data, the head was crudely chosen to be the rightmost mention token, and the mention type was automatically detected. We will not assume any other information to be present in the data beyond the text itself. In particular, unlike much related work, we do not assume gold named entity recognition (NER) labels; indeed we do not assume observed NER labels or POS tags at all. Our primary performance metric will be the MUC F_1 measure (Vilain et al., 1995), commonly used to evaluate coreference systems on a within-document basis. Since our system relies on sampling, all results are averaged over five random runs.

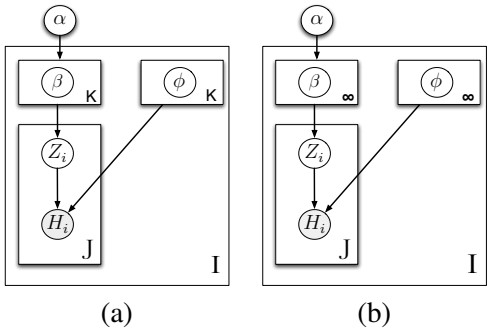


Figure 1: Graphical model depiction of document level entity models described in sections 3.1 and 3.2 respectively. The shaded nodes indicate observed variables.

3 Coreference Resolution Models

In this section, we present a sequence of generative coreference resolution models for document corpora. All are essentially mixture models, where the mixture components correspond to entities. As far as notation, we assume a collection of I documents, each with J_i mentions. We use random variables Z to refer to (indices of) entities. We will use ϕ_z to denote the parameters for an entity z, and $\boldsymbol{\phi}$ to refer to the concatenation of all such ϕ_z. X will refer somewhat loosely to the collection of variables associated with a mention in our model (such as the head or gender). We will be explicit about X and ϕ_z shortly.

Our goal will be to find the setting of the entity indices which maximize the posterior probability:

$$\begin{aligned}\mathbf{Z}^* &= \arg\max_{\mathbf{Z}} P(\mathbf{Z}|\mathbf{X}) = \arg\max_{\mathbf{Z}} P(\mathbf{Z},\mathbf{X}) \\ &= \arg\max_{\mathbf{Z}} \int P(\mathbf{Z},\mathbf{X},\boldsymbol{\phi})\, dP(\boldsymbol{\phi})\end{aligned}$$

where $\mathbf{Z}$, $\mathbf{X}$, and $\boldsymbol{\phi}$ denote all the entity indices, observed values, and parameters of the model. Note that we take a Bayesian approach in which all parameters are integrated out (or sampled). The inference task is thus primarily a search problem over the index labels $\mathbf{Z}$.

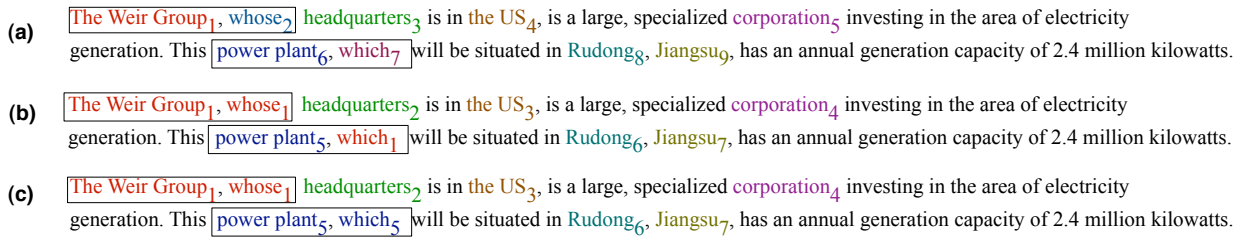


Figure 2: Example output from various models. The output from (a) is from the infinite mixture model of section 3.2. It incorrectly labels both boxed cases of anaphora. The output from (b) uses the pronoun head model of section 3.3. It correctly labels the first case of anaphora but incorrectly labels the second pronominal as being coreferent with the dominant document entity *The Weir Group*. This error is fixed by adding the salience feature component from section 3.4 as can be seen in (c).

3.1 A Finite Mixture Model

Our first, overly simplistic, corpus model is the standard finite mixture of multinomials shown in figure 1(a). In this model, each document is independent save for some global hyperparameters. Inside each document, there is a finite mixture model with a fixed number K of components. The distribution β over components (entities) is a draw from a symmetric Dirichlet distribution with concentration α. For each mention in the document, we choose a component (an entity index) z from β. Entity z is then associated with a multinomial emission distribution over head words with parameters ϕ_Z^h, which are drawn from a symmetric Dirichlet over possible mention heads with concentration λ_H.[2] Note that here the X for a mention consists only of the mention head H.

As we enrich our models, we simultaneously develop an accompanying Gibbs sampling procedure to obtain samples from $P(\mathbf{Z}|\mathbf{X})$.[3] For now, all heads H are observed and all parameters (β and $\boldsymbol{\phi}$) can be integrated out analytically: for details see Teh et al. (2006). The only sampling is for the values of $Z_{i,j}$, the entity index of mention j in document i. The relevant conditional distribution is:[4]

$$P(Z_{i,j}|\mathbf{Z}^{-i,j}, \mathbf{H}) \propto P(Z_{i,j}|\mathbf{Z}^{-i,j})P(H_{i,j}|\mathbf{Z}, \mathbf{H}^{-i,j})$$

where $H_{i,j}$ is the head of mention j in document i. Expanding each term, we have the contribution of the prior:

$$P(Z_{i,j} = z|\mathbf{Z}^{-i,j}) \propto n_z + \alpha$$

where n_z is the number of elements of $\mathbf{Z}^{-i,j}$ with entity index z. Similarly we have for the contribution of the emissions:

$$P(H_{i,j} = h|\mathbf{Z}, \mathbf{H}^{-i,j}) \propto n_{h,z} + \lambda_H$$

where $n_{h,z}$ is the number of times we have seen head h associated with entity index z in $(\mathbf{Z}, \mathbf{H}^{-i,j})$.

3.2 An Infinite Mixture Model

A clear drawback of the finite mixture model is the requirement that we specify a priori a number of entities K for a document. We would like our model to select K in an effective, principled way. A mechanism for doing so is to replace the finite Dirichlet prior on β with the non-parametric Dirichlet process (DP) prior (Ferguson, 1973).[5] Doing so gives the model in figure 1(b). Note that we now list an infinite number of mixture components in this model since there can be an unbounded number of entities. Rather than a finite β with a symmetric Dirichlet distribution, in which draws tend to have balanced clusters, we now have an infinite β. However, most draws will have weights which decay exponentially quickly in the prior (though not necessarily in the posterior). Therefore, there is a natural penalty for each cluster which is actually used.

With $\mathbf{Z}$ observed during sampling, we can integrate out β and calculate $P(Z_{i,j}|\mathbf{Z}^{-i,j})$ analytically, using the Chinese restaurant process representation:

$$P(Z_{i,j} = z|\mathbf{Z}^{-i,j}) \propto \begin{cases} \alpha, & \text{if } z = z_{new} \\ n_z, & \text{otherwise} \end{cases} \quad (1)$$

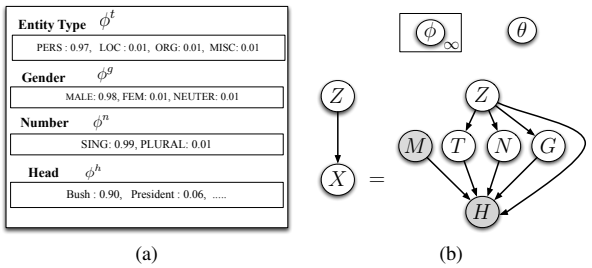
where z_{new} is a new entity index not used in $\mathbf{Z}^{-i,j}$ and n_z is the number of mentions that have entity index z. Aside from this change, sampling is identical

[2] In general, we will use a subscripted λ to indicate concentration for finite Dirichlet distributions. Unless otherwise specified, λ concentration parameters will be set to e^{-4} and omitted from diagrams.

[3] One could use the EM algorithm with this model, but EM will not extend effectively to the subsequent models.

[4] Here, $\mathbf{Z}^{-i,j}$ denotes $\mathbf{Z} - \{Z_{i,j}\}$

[5] We do not give a detailed presentation of the Dirichlet process here, but see Teh et al. (2006) for a presentation.



(a) (b)

Figure 3: (a) An entity and its parameters. (b)The head model described in section 3.3. The shaded nodes indicate observed variables. The mention type determines which set of parents are used. The dependence of mention variable on entity parameters $\boldsymbol{\phi}$ and pronoun head model $\boldsymbol{\theta}$ is omitted.

to the finite mixture case, though with the number of clusters actually occupied in each sample drifting upwards or downwards.

This model yielded a 54.5 F_1 on our development data.[6] This model is, however, hopelessly crude, capturing nothing of the structure of coreference. Its largest empirical problem is that, unsurprisingly, pronoun mentions such as *he* are given their own clusters, not labeled as coreferent with any non-pronominal mention (see figure 2(a)).

3.3 Pronoun Head Model

While an entity-specific multinomial distribution over heads makes sense for proper, and some nominal, mention heads, it does not make sense to generate pronominal mentions this same way. I.e., all entities can be referred to by generic pronouns, the choice of which depends on entity properties such as gender, not the specific entity.

We therefore enrich an entity's parameters ϕ to contain not only a distribution over lexical heads ϕ^h, but also distributions (ϕ^t, ϕ^g, ϕ^n) over properties, where ϕ^t parametrizes a distribution over entity types (PER, LOC, ORG, MISC), and ϕ^g for gender (MALE, FEMALE, NEUTER), and ϕ^n for number (SG, PL).[7] We assume each of these property distributions is drawn from a symmetric Dirichlet distribution with small concentration parameter in order to encourage a peaked posterior distribution.

[6] See section 4 for inference details.

[7] It might seem that entities should simply have, for example, a gender g rather than a distribution over genders ϕ^g. There are two reasons to adopt the softer approach. First, one can rationalize it in principle, for entities like cars or ships whose grammatical gender is not deterministic. However, the real reason is that inference is simplified. In any event, we found these property distributions to be highly determinized in the posterior.

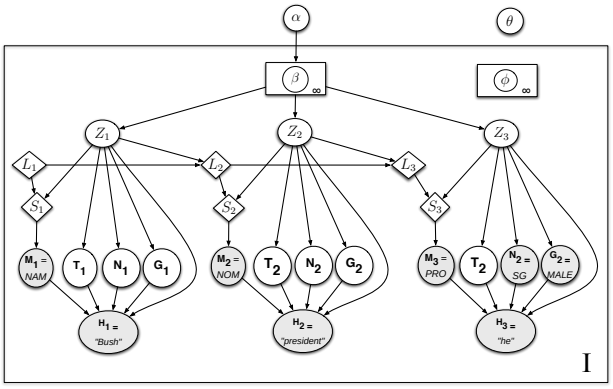


Figure 4: Coreference model at the document level with entity properties as well salience lists used for mention type distributions. The diamond nodes indicate deterministic functions. Shaded nodes indicate observed variables. Although it appears that each mention head node has many parents, for a given mention type, the mention head depends on only a small subset. Dependencies involving parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ are omitted.

Previously, when an entity z generated a mention, it drew a head word from ϕ^h_z. It now undergoes a more complex and structured process. It first draws an entity type T, a gender G, a number N from the distributions ϕ^t, ϕ^g, and ϕ^n, respectively. Once the properties are fetched, a mention type M is chosen (*proper*, *nominal*, *pronoun*), according to a global multinomial (again with a symmetric Dirichlet prior and parameter λ_M). This corresponds to the (temporary) assumption that the speaker makes a random i.i.d. choice for the type of each mention.

Our head model will then generate a head, conditioning on the entity, its properties, and the mention type, as shown in figure 3(b). If M is not a pronoun, the head is drawn directly from the entity head multinomial with parameters ϕ^h_z. Otherwise, it is drawn based on a global pronoun head distribution, conditioning on the entity properties and parametrized by $\boldsymbol{\theta}$. Formally, it is given by:

$$P(H|Z,T,G,N,M,\boldsymbol{\phi},\boldsymbol{\theta}) = \begin{cases} P(H|T,G,N,\boldsymbol{\theta}), & \text{if } M = \text{PRO} \\ P(H|\phi^h_Z), & \text{otherwise} \end{cases}$$

Although we can observe the number and gender draws for some mentions, like personal pronouns, there are some for which properties aren't observed (e.g., *it*). Because the entity property draws are not (all) observed, we must now sample the unobserved ones as well as the entity indices Z. For instance, we could sample

Salience Feature	Pronoun	Proper	Nominal
TOP	0.75	0.17	0.08
HIGH	0.55	0.28	0.17
MID	0.39	0.40	0.21
LOW	0.20	0.45	0.35
NONE	0.00	0.88	0.12

Table 1: Posterior distribution of mention type given salience by bucketing entity activation rank. Pronouns are preferred for entities which have high salience and non-pronominal mentions are preferred for inactive entities.

$T_{i,j}$, the entity type of pronominal mention j in document i, using, $P(T_{i,j}|\mathbf{Z}, \mathbf{N}, \mathbf{G}, \mathbf{H}, \mathbf{T}^{-i,j}) \propto P(T_{i,j}|\mathbf{Z})P(H_{i,j}|\mathbf{T}, \mathbf{N}, \mathbf{G}, \mathbf{H})$, where the posterior distributions on the right hand side are straightforward because the parameter priors are all finite Dirichlet. Sampling G and N are identical.

Of course we have prior knowledge about the relationship between entity type and pronoun head choice. For example, we expect that *he* is used for mentions with $T = \text{PERSON}$. In general, we assume that for each pronominal head we have a list of compatible entity types, which we encode via the prior on $\boldsymbol{\theta}$. We assume $\boldsymbol{\theta}$ is drawn from a Dirichlet distribution where each pronoun head is given a synthetic count of $(1 + \lambda_P)$ for each (t, g, n) where t is compatible with the pronoun and given λ_P otherwise. So, while it will be possible in the posterior to use *he* to refer to a non-person, it will be biased towards being used with persons.

This model gives substantially improved predictions: 64.1 F_1 on our development data. As can be seen in figure 2(b), this model does correct the systematic problem of pronouns being considered their own entities. However, it still does not have a preference for associating pronominal references to entities which are in any way local.

3.4 Adding Salience

We would like our model to capture how mention types are generated for a given entity in a robust and somewhat language independent way. The choice of entities may reasonably be considered to be independent given the mixing weights β, but how we *realize* an entity is strongly dependent on context (Ge et al., 1998).

In order to capture this in our model, we enrich it as shown in figure 4. As we proceed through a document, generating entities and their mentions, we maintain a list of the active entities and their *saliences*, or activity scores. Every time an entity is mentioned, we increment its activity score by 1, and every time we move to generate the next mention, all activity scores decay by a constant factor of 0.5. This gives rise to an ordered list of entity activations, L, where the rank of an entity decays exponentially as new mentions are generated. We call this list a *salience list*. Given a salience list, L, each possible entity z has some rank on this list. We discretize these ranks into five buckets S: TOP (1), HIGH (2-3), MID (4-6), LOW (7+), and NONE. Given the entity choices $\mathbf{Z}$, both the list L and buckets S are deterministic (see figure 4). We assume that the mention type M is conditioned on S as shown in figure 4.

We note that correctly sampling an entity now requires that we incorporate terms for how a change will affect all future salience values. This changes our sampling equation for existing entities:

$$P(Z_{i,j} = z|\mathbf{Z}^{-i,j}) \propto n_z \prod_{j' \geq j} P(M_{i,j'}|S_{i,j'}, \mathbf{Z}) \quad (2)$$

where the product ranges over future mentions in the document and $S_{i,j'}$ is the value of future salience feature given the setting of all entities, including setting the current entity $Z_{i,j}$ to z. A similar equation holds for sampling a new entity. Note that, as discussed below, this full product can be truncated as an approximation.

This model gives a 71.5 F_1 on our development data. Table 1 shows the posterior distribution of the mention type given the salience feature. This model fixes many anaphora errors and in particular fixes the second anaphora error in figure 2(c).

3.5 Cross Document Coreference

One advantage of a fully generative approach is that we can allow entities to be shared between documents in a principled way, giving us the capacity to do cross-document coreference. Moreover, sharing across documents pools information about the properties of an entity across documents.

We can easily link entities across a corpus by assuming that the pool of entities is global, with global mixing weights β_0 drawn from a DP prior with concentration parameter γ. Each document uses

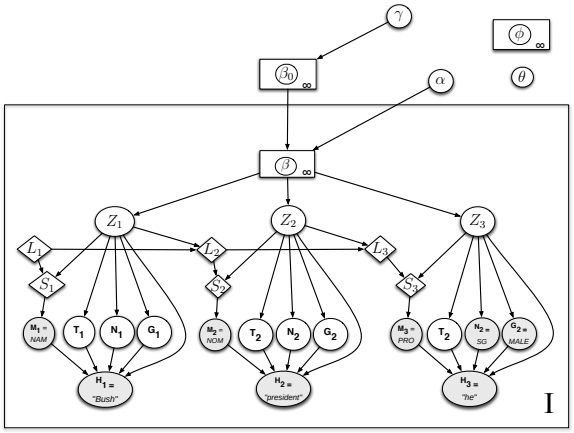


Figure 5: Graphical depiction of the HDP coreference model described in section 3.5. The dependencies between the global entity parameters ϕ and pronoun head parameters θ on the mention observations are not depicted.

the same global entities, but each has a document-specific distribution β_i drawn from a DP centered on β_0 with concentration parameter α. Up to the point where entities are chosen, this formulation follows the basic hierarchical Dirichlet process prior of Teh et al. (2006). Once the entities are chosen, our model for the realization of the mentions is as before. This model is depicted graphically in figure 5.

Although it is possible to integrate out β_0 as we did the individual β_i, we instead choose for efficiency and simplicity to sample the global mixture distribution β_0 from the posterior distribution $P(\beta_0|\mathbf{Z})$.[8] The mention generation terms in the model and sampler are unchanged.

In the full hierarchical model, our equation (1) for sampling entities, ignoring the salience component of section 3.4, becomes:

$$P(Z_{i,j} = z|\mathbf{Z}^{-i,j}, \beta_0) \propto \begin{cases} \alpha\beta_0^u, & \text{if } z = z_{new} \\ n_z + \alpha\beta_0^z, & \text{otherwise} \end{cases}$$

where β_0^z is the probability of the entity z under the sampled global entity distribution and β_0^u is the unknown component mass of this distribution.

The HDP layer of sharing improves the model's predictions to 72.5 F_1 on our development data. We should emphasize that our evaluation is of course per-document and does not reflect cross-document coreference decisions, only the gains through cross-document sharing (see section 6.2).

[8] We do not give the details here; see Teh et al. (2006) for details on how to implement this component of the sampler (called "direct assignment" in that reference).

4 Inference Details

Up until now, we've discussed Gibbs sampling, but we are not interested in sampling from the posterior $P(\mathbf{Z}|\mathbf{X})$, but in finding its mode. Instead of sampling directly from the posterior distribution, we instead sample entities proportionally to exponentiated entity posteriors. The exponent is given by $\exp \frac{ci}{k-1}$, where i is the current round number (starting at $i = 0$), $c = 1.5$ and $k = 20$ is the total number of sampling epochs. This slowly raises the posterior exponent from 1.0 to e^c. In our experiments, we found this procedure to outperform simulated annealing. We also found sampling the T, G, and N variables to be particularly inefficient, so instead we maintain soft counts over each of these variables and use these in place of a hard sampling scheme. We also found that correctly accounting for the future impact of salience changes to be particularly inefficient. However, ignoring those terms entirely made negligible difference in final accuracy.[9]

5 Final Experiments

We present our final experiments using the full model developed in section 3. As in section 3, we use true mention boundaries and evaluate using the MUC F_1 measure (Vilain et al., 1995). All hyperparameters were tuned on the development set only. The document concentration parameter α was set by taking a constant proportion of the average number of *mentions* in a document across the corpus. This number was chosen to minimize the squared error between the number of proposed entities and true entities in a document. It was not tuned to maximize the F_1 measure. A coefficient of 0.4 was chosen. The global concentration coefficient γ was chosen to be a constant proportion of αM, where M is the number of *documents* in the corpus. We found 0.15 to be a good value using the same least-square procedure. The values for these coefficients were not changed for the experiments on the test sets.

5.1 MUC-6

Our main evaluation is on the standard MUC-6 formal test set.[10] The standard experimental setup for

[9] This corresponds to truncating equation (2) at $j' = j$.

[10] Since the MUC data is not annotated with mention types, we automatically detect this information in the same way as Luo

Dataset	Num Docs.	Prec.	Recall	F_1
MUC-6	60	80.8	52.8	63.9
+DRYRUN-TRAIN	251	79.1	59.7	68.0
+ENGLISH-NWIRE	381	80.4	62.4	70.3

(a)

Dataset	Prec.	Recall	F_1
ENGLISH-NWIRE	66.7	62.3	64.2
ENGLISH-BNEWS	63.2	61.3	62.3
CHINESE-NWIRE	71.6	63.3	67.2
CHINESE-BNEWS	71.2	61.8	66.2

(b)

Table 2: Formal Results: Our system evaluated using the MUC model theoretic measure Vilain et al. (1995). The table in (a) is our performance on the thirty document MUC-6 formal test set with increasing amounts of training data. In all cases for the table, we are evaluating on the same thirty document test set which is included in our training set, since our system in unsupervised. The table in (b) is our performance on the ACE 2004 training sets.

this data is a 30/30 document train/test split. Training our system on all 60 documents of the training and test set (as this is in an unsupervised system, the unlabeled test documents are present at training time), but evaluating only on the test documents, gave 63.9 F_1 and is labeled MUC-6 in table 2(a).

One advantage of an unsupervised approach is that we can easily utilize more data when learning a model. We demonstrate the effectiveness of this fact by evaluating on the MUC-6 test documents with increasing amounts of unannotated training data. We first added the 191 documents from the MUC-6 dryrun training set (which were not part of the training data for official MUC-6 evaluation). This model gave 68.0 F_1 and is labeled +DRYRUN-TRAIN in table 2(a). We then added the ACE ENGLISH-NWIRE training data, which is from a different corpora than the MUC-6 test set and from a different time period. This model gave 70.3 F_1 and is labeled +ENGLISH-NWIRE in table 2(a).

Our results on this test set are surprisingly comparable to, though slightly lower than, some recent supervised systems. McCallum and Wellner (2004) report 73.4 F_1 on the formal MUC-6 test set, which is reasonably close to our best MUC-6 number of 70.3 F_1. McCallum and Wellner (2004) also report a much lower 91.6 F_1 on only proper nouns mentions. Our system achieves a 89.8 F_1 when evaluation is restricted to only proper mentions.[11] The

HEAD	ENT TYPE	GENDER	NUMBER
Bush: 1.0	PERS	MALE	SG
AP: 1.0	ORG	NEUTER	PL
viacom: 0.64, *company*: 0.36	ORG	NEUTER	SG
teamsters: 0.22, *union*: 0.78,	MISC	NEUTER	PL

Table 3: Frequent entities occurring across documents along with head distribution and mode of property distributions.

closest comparable unsupervised system is Cardie and Wagstaff (1999) who use pairwise NP distances to cluster document mentions. They report a 53.6 F_1 on MUC6 when tuning distance metric weights to maximize F_1 on the development set.

5.2 ACE 2004

We also performed experiments on ACE 2004 data. Due to licensing restrictions, we did not have access to the ACE 2004 formal development and test sets, and so the results presented are on the training sets.

We report results on the newswire section (NWIRE in table 2b) and the broadcast news section (BNEWS in table 2b). These datasets include the *prenominal* mention type, which is not present in the MUC-6 data. We treated prenominals analogously to the treatment of proper and nominal mentions.

We also tested our system on the Chinese newswire and broadcast news sections of the ACE 2004 training sets. Our relatively higher performance on Chinese compared to English is perhaps due to the lack of prenominal mentions in the Chinese data, as well as the presence of fewer pronouns compared to English.

Our ACE results are difficult to compare exactly to previous work because we did not have access to the restricted formal test set. However, we can perform a rough comparison between our results on the training data (without coreference annotation) to supervised work which has used the same training data (with coreference annotation) and evaluated on the formal test set. Denis and Baldridge (2007) re-

et al. (2004). A mention is proper if it is annotated with NER information. It is a pronoun if the head is on the list of English pronouns. Otherwise, it is a nominal mention. Note we do not use the NER information for any purpose but determining whether the mention is proper.

[11] The best results we know on the MUC-6 test set using the standard setting are due to Luo et al. (2004) who report a 81.3 F_1 (much higher than others). However, it is not clear this is a comparable number, due to the apparent use of gold NER features, which provide a strong clue to coreference. Regardless, it is unsurprising that their system, which has many rich features, would outperform ours.

port 67.1 F_1 and 69.2 F_1 on the English NWIRE and BNEWS respectively using true mention boundaries. While our system underperforms the supervised systems, its accuracy is nonetheless promising.

6 Discussion

6.1 Error Analysis

The largest source of error in our system is between coreferent proper and nominal mentions. The most common examples of this kind of error are appositive usages e.g. *George W. Bush, president of the US, visited Idaho.* Another error of this sort can be seen in figure 2, where the *corporation* mention is not labeled coreferent with the *The Weir Group* mention. Examples such as these illustrate the regular (at least in newswire) phenomenon that nominal mentions are used with informative intent, even when the entity is salient and a pronoun could have been used unambiguously. This aspect of nominal mentions is entirely unmodeled in our system.

6.2 Global Coreference

Since we do not have labeled cross-document coreference data, we cannot evaluate our system's cross-document performance quantitatively. However, in addition to observing the within-document gains from sharing shown in section 3, we can manually inspect the most frequently occurring entities in our corpora. Table 3 shows some of the most frequently occurring entities across the English ACE NWIRE corpus. Note that *Bush* is the most frequent entity, though his (and others') nominal cluster *president* is mistakenly its own entity. Merging of proper and nominal clusters does occur as can be seen in table 3.

6.3 Unsupervised NER

We can use our model to for unsupervised NER tagging: for each proper mention, assign the mode of the generating entity's distribution over entity types. Note that in our model the only way an entity becomes associated with an entity type is by the pronouns used to refer to it.[12] If we evaluate our system as an unsupervised NER tagger for the proper mentions in the MUC-6 test set, it yields a per-label accuracy of 61.2% (on MUC labels). Although nowhere near the performance of state-of-the-art systems, this result beats a simple baseline of always guessing PERSON (the most common entity type), which yields 46.4%. This result is interesting given that the model was not developed for the purpose of inferring entity types whatsoever.

[12] Ge et al. (1998) exploit a similar idea to assign gender to proper mentions.

7 Conclusion

We have presented a novel, unsupervised approach to coreference resolution: global entities are shared across documents, the number of entities is determined by the model, and mentions are generated by a sequential salience model and a model of pronoun-entity association. Although our system does not perform quite as well as state-of-the-art supervised systems, its performance is in the same general range, despite the system being unsupervised.

References

I. Bhattacharya and L. Getoor. 2006. A latent dirichlet model for unsupervised entity resolution. *SIAM conference on data mining*.

Claire Cardie and Kiri Wagstaff. 1999. Noun phrase coreference as clustering. *EMNLP*.

Hal Daume and Daniel Marcu. 2005. A Bayesian model for supervised clustering with the Dirichlet process prior. *JMLR*.

Pascal Denis and Jason Baldridge. 2007. Global, joint determination of anaphoricity and coreference resolution using integer programming. *HLT-NAACL*.

Thomas Ferguson. 1973. A bayesian analysis of some nonparametric problems. *Annals of Statistics*.

Niyu Ge, John Hale, and Eugene Charniak. 1998. A statistical approach to anaphora resolution. *Sixth Workshop on Very Large Corpora*.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the bell tree. *ACL*.

Andrew McCallum and Ben Wellner. 2004. Conditional models of identity uncertainty with application to noun coreference. *NIPS*.

Brian Milch, Bhaskara Marthi, Stuart Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. 2005. Blog: Probabilistic models with unknown objects. *IJCAI*.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. *ACL*.

NIST. 2004. The ACE evaluation plan.

W. Soon, H. Ng, and D. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*.

Yee Whye Teh, Michael Jordan, Matthew Beal, and David Blei. 2006. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. *MUC-6*.

Pivot Language Approach for Phrase-Based Statistical Machine Translation

Hua Wu and Haifeng Wang
Toshiba (China) Research and Development Center
5/F., Tower W2, Oriental Plaza, No.1, East Chang An Ave., Dong Cheng District
Beijing, 100738, China
{wuhua,wanghaifeng}@rdc.toshiba.com.cn

Abstract

This paper proposes a novel method for phrase-based statistical machine translation by using pivot language. To conduct translation between languages L_f and L_e with a small bilingual corpus, we bring in a third language L_p, which is named the *pivot* language. For L_f-L_p and L_p-L_e, there exist large bilingual corpora. Using only L_f-L_p and L_p-L_e bilingual corpora, we can build a translation model for L_f-L_e. The advantage of this method lies in that we can perform translation between L_f and L_e even if there is no bilingual corpus available for this language pair. Using BLEU as a metric, our pivot language method achieves an absolute improvement of 0.06 (22.13% relative) as compared with the model directly trained with 5,000 L_f-L_e sentence pairs for French-Spanish translation. Moreover, with a small L_f-L_e bilingual corpus available, our method can further improve the translation quality by using the additional L_f-L_p and L_p-L_e bilingual corpora.

1 Introduction

For statistical machine translation (SMT), phrase-based methods (Koehn et al., 2003; Och and Ney, 2004) and syntax-based methods (Wu, 1997; Alshawi et al. 2000; Yamada and Knignt, 2001; Melamed, 2004; Chiang, 2005; Quick et al., 2005; Mellebeek et al., 2006) outperform word-based methods (Brown et al., 1993). These methods need large bilingual corpora. However, for some languages pairs, only a small bilingual corpus is available, which will degrade the performance of statistical translation systems.

To solve this problem, this paper proposes a novel method for phrase-based SMT by using a pivot language. To perform translation between languages L_f and L_e, we bring in a pivot language L_p, for which there exist large bilingual corpora for language pairs L_f-L_p and L_p-L_e. With the L_f-L_p and L_p-L_e bilingual corpora, we can build a translation model for L_f-L_e by using L_p as the pivot language. We name the translation model *pivot* model. The advantage of this method lies in that we can conduct translation between L_f and L_e even if there is no bilingual corpus available for this language pair. Moreover, if a small corpus is available for L_f-L_e, we build another translation model, which is named *standard* model. Then, we build an *interpolated* model by performing linear interpolation on the standard model and the pivot model. Thus, the interpolated model can employ both the small L_f-L_e corpus and the large L_f-L_p and L_p-L_e corpora.

We perform experiments on the Europarl corpus (Koehn, 2005). Using BLEU (Papineni et al., 2002) as a metric, our method achieves an absolute improvement of 0.06 (22.13% relative) as compared with the standard model trained with 5,000 L_f-L_e sentence pairs for French-Spanish translation. The translation quality is comparable with that of the model trained with a bilingual corpus of 30,000 L_f-L_e sentence pairs. Moreover, translation quality is further boosted by using both the small L_f-L_e bilingual corpus and the large L_f-L_p and L_p-L_e corpora.

Experimental results on Chinese-Japanese translation also indicate that our method achieves satisfactory results using English as the pivot language.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 856–863,
Prague, Czech Republic, June 2007.

The remainder of this paper is organized as follows. In section 2, we describe the related work. Section 3 briefly introduces phrase-based SMT. Section 4 and Section 5 describes our method for phrase-based SMT using pivot language. We describe the experimental results in sections 6 and 7. Lastly, we conclude in section 8.

2 Related Work

Our method is mainly related to two kinds of methods: those using pivot language and those using a small bilingual corpus or scarce resources.

For the first kind, pivot languages are employed to translate queries in cross-language information retrieval (CLIR) (Gollins and Sanderson, 2001; Kishida and Kando, 2003). These methods only used the available dictionaries to perform word by word translation. In addition, NTCIR 4 workshop organized a shared task for CLIR using pivot language. Machine translation systems are used to translate queries into pivot language sentences, and then into target sentences (Sakai et al., 2004).

Callison-Burch et al. (2006) used pivot languages for paraphrase extraction to handle the unseen phrases for phrase-based SMT. Borin (2000) and Wang et al. (2006) used pivot languages to improve word alignment. Borin (2000) used multilingual corpora to increase alignment coverage. Wang et al. (2006) induced alignment models by using two additional bilingual corpora to improve word alignment quality. Pivot Language methods were also used for translation dictionary induction (Schafer and Yarowsky, 2002), word sense disambiguation (Diab and Resnik, 2002), and so on.

For the second kind, Niessen and Ney (2004) used morpho-syntactic information for translation between language pairs with scarce resources. Vandeghinste et al. (2006) used translation dictionaries and shallow analysis tools for translation between the language pair with low resources. A shared task on word alignment was organized as part of the ACL 2005 Workshop on Building and Using Parallel Texts (Martin et al., 2005). This task focused on languages with scarce resources. For the subtask of unlimited resources, some researchers (Aswani and Gaizauskas, 2005; Lopez and Resnik, 2005; Tufis et al., 2005) used language-dependent resources such as dictionary, thesaurus, and dependency parser to improve word alignment results.

In this paper, we address the translation problem for language pairs with scarce resources by bringing in a pivot language, via which we can make use of large bilingual corpora. Our method does not need language-dependent resources or deep linguistic processing. Thus, the method is easy to be adapted to any language pair where a pivot language and corresponding large bilingual corpora are available.

3 Phrase-Based SMT

According to the translation model presented in (Koehn et al., 2003), given a source sentence $\mathbf{f}$, the best target translation $\mathbf{e}_{\text{best}}$ can be obtained according to the following model

$$\begin{aligned}\mathbf{e}_{\text{best}} &= \arg\max_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}) \\ &= \arg\max_{\mathbf{e}} p(\mathbf{f}|\mathbf{e}) p_{\text{LM}}(\mathbf{e}) \omega^{\text{length}(\mathbf{e})}\end{aligned} \quad (1)$$

Where the translation model $p(\mathbf{f}|\mathbf{e})$ can be decomposed into

$$\begin{aligned}&p(\bar{f}_1^I | \bar{e}_1^I) \\ &= \prod_{i=1}^{I} \phi(\bar{f}_i | \bar{e}_i) d(a_i - b_{i-1}) p_{\text{w}}(\bar{f}_i | \bar{e}_i, a)^{\lambda}\end{aligned} \quad (2)$$

Where $\phi(\bar{f}_i | \bar{e}_i)$ and $d(a_i - b_{i-1})$ denote phrase translation probability and distortion probability, respectively. $p_{\text{w}}(\bar{f}_i | \bar{e}_i, a)$ is the lexical weight, and λ is the strength of the lexical weight.

4 Phrase-Based SMT Via Pivot Language

This section will introduce the method that performs phrase-based SMT for the language pair L_f-L_e by using the two bilingual corpora of L_f-L_p and L_p-L_e. With the two additional bilingual corpora, we train two translation models for L_f-L_p and L_p-L_e, respectively. Based on these two models, we build a pivot translation model for L_f-L_e, with L_p as a pivot language.

According to equation (2), the phrase translation probability and the lexical weight are language dependent. We will introduce them in sections 4.1 and 4.2, respectively.

4.1 Phrase Translation Probability

Using the L_f-L_p and L_p-L_e bilingual corpora, we train two phrase translation probabilities

$\phi(\bar{f}_i \mid \bar{p}_i)$ and $\phi(\bar{p}_i \mid \bar{e}_i)$, where $\bar{p}_i$ is the phrase in the pivot language L_p. Given the phrase translation probabilities $\phi(\bar{f}_i \mid \bar{p}_i)$ and $\phi(\bar{p}_i \mid \bar{e}_i)$, we obtain the phrase translation probability $\phi(\bar{f}_i \mid \bar{e}_i)$ according to the following model.

$$\phi(\bar{f}_i \mid \bar{e}_i) = \sum_{\bar{p}_i} \phi(\bar{f}_i \mid \bar{p}_i, \bar{e}_i)\phi(\bar{p}_i \mid \bar{e}_i) \quad (3)$$

The phrase translation probability $\phi(\bar{f}_i \mid \bar{p}_i, \bar{e}_i)$ does not depend on the phase $\bar{e}_i$ in the language L_e, since it is estimated from the L_f-L_p bilingual corpus. Thus, equation (3) can be rewritten as

$$\phi(\bar{f}_i \mid \bar{e}_i) = \sum_{\bar{p}_i} \phi(\bar{f}_i \mid \bar{p}_i)\phi(\bar{p}_i \mid \bar{e}_i) \quad (4)$$

4.2 Lexical Weight

Given a phrase pair $(\bar{f}, \bar{e})$ and a word alignment a between the source word positions $i = 1,...,n$ and the target word positions $j = 1,...,m$, the lexical weight can be estimated according to the following method (Koehn et al., 2003).

$$p_w(\bar{f} \mid \bar{e}, a) = \prod_{i=1}^{n} \frac{1}{|j \mid (i,j) \in a|} \sum_{\forall (i,j) \in a} w(f_i \mid e_j) \quad (5)$$

In order to estimate the lexical weight, we first need to obtain the alignment information a between the two phrases $\bar{f}$ and $\bar{e}$, and then estimate the lexical translation probability $w(f \mid e)$ according to the alignment information. The alignment information of the phrase pair $(\bar{f}, \bar{e})$ can be induced from the two phrase pairs $(\bar{f}, \bar{p})$ and $(\bar{p}, \bar{e})$.

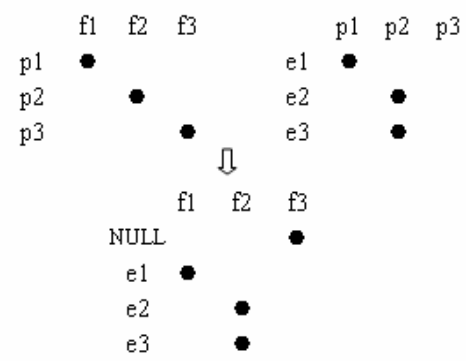


Figure 1. Alignment Information Induction

Let a_1 and a_2 represent the word alignment information inside the phrase pairs $(\bar{f}, \bar{p})$ and $(\bar{p}, \bar{e})$ respectively, then the alignment information a inside $(\bar{f}, \bar{e})$ can be obtained as shown in (6). An example is shown in Figure 1.

$$a = \{(f, e) \mid \exists p : (f, p) \in a_1 \ \& \ (p, e) \in a_2\} \quad (6)$$

With the induced alignment information, this paper proposes a method to estimate the probability directly from the induced phrase pairs. We name this method *phrase* method. If we use K to denote the number of the induced phrase pairs, we estimate the co-occurring frequency of the word pair (f, e) according to the following model.

$$count(f, e) = \sum_{k=1}^{K} \phi_k(\bar{f} \mid \bar{e}) \sum_{i=1}^{n} \delta(f, f_i)\delta(e, e_{a_i}) \quad (7)$$

Where $\phi_k(\bar{f} \mid \bar{e})$ is the phrase translation probability for phrase pair k. $\delta(x, y) = 1$ if $x = y$; otherwise, $\delta(x, y) = 0$. Thus, lexical translation probability can be estimated as in (8).

$$w(f \mid e) = \frac{count(f, e)}{\sum_{f'} count(f', e)} \quad (8)$$

We also estimate the lexical translation probability $w(f \mid e)$ using the method described in (Wang et al., 2006), which is shown in (9). We named it *word* method in this paper.

$$w(f \mid e) = \sum_{p} w(f \mid p)w(p \mid e)sim(f, e; p) \quad (9)$$

Where $w(f \mid p)$ and $w(p \mid e)$ are two lexical probabilities, and $sim(f, e; p)$ is the cross-language word similarity.

5 Interpolated Model

If we have a small L_f-L_e bilingual corpus, we can employ this corpus to estimate a translation model as described in section 3. However, this model may perform poorly due to the sparseness of the data. In order to improve its performance, we can employ the additional L_f-L_p and L_p-L_e bilingual corpora. Moreover, we can use more than one pivot language to improve the translation performance if the corresponding bilingual corpora exist. Different pivot languages may catch different linguistic phe-

nomena, and improve translation quality for the desired language pair L_f-L_e in different ways.

If we include n pivot languages, n pivot models can be estimated using the method as described in section 4. In order to combine these n pivot models with the standard model trained with the L_f-L_e corpus, we use the linear interpolation method. The phrase translation probability and the lexical weight are estimated as shown in (10) and (11), respectively.

$$\phi(\bar{f} \mid \bar{e}) = \sum_{i=0}^{n} \alpha_i \phi_i(\bar{f} \mid \bar{e}) \quad (10)$$

$$p_w(\bar{f} \mid \bar{e}, a) = \sum_{i=0}^{n} \beta_i p_{w,i}(\bar{f} \mid \bar{e}, a) \quad (11)$$

Where $\phi_0(\bar{f} \mid \bar{e})$ and $p_{w,0}(\bar{f} \mid \bar{e}, a)$ denote the phrase translation probability and lexical weight trained with the L_f-L_e bilingual corpus, respectively. $\phi_i(\bar{f} \mid \bar{e})$ and $p_{w,i}(\bar{f} \mid \bar{e}, a)$ $(i = 1,...,n)$ are the phrase translation probability and lexical weight estimated by using the pivot languages. α_i and β_i are the interpolation coefficients.

6 Experiments on the Europarl Corpus

6.1 Data

A shared task to evaluate machine translation performance was organized as part of the NAACL/HLT 2006 Workshop on Statistical Machine Translation (Koehn and Monz, 2006). The shared task used the Europarl corpus (Koehn, 2005), in which four languages are involved: English, French, Spanish, and German. The shared task performed translation between English and the other three languages. In our work, we perform translation from French to the other three languages. We select French to Spanish and French to German translation that are not in the shared task because we want to use English as the pivot language. In general, for most of the languages, there exist bilingual corpora between these languages and English since English is an internationally used language.

Table 1 shows the information about the bilingual training data. In the table, "Fr", "En", "Es", and "De" denotes "French", "English", "Spanish", and "German", respectively. For the language pairs L_f-L_e not including English, the bilingual corpus is extracted from L_f-English and English-L_e since Europarl corpus is a multilingual corpus.

Language Pairs	Sentence Pairs	Source Words	Target Words
Fr-En	688,031	15,323,737	13,808,104
Fr-Es	640,661	14,148,926	13,134,411
Fr-De	639,693	14,215,058	12,155,876
Es-En	730,740	15,676,710	15,222,105
De-En	751,088	15,256,793	16,052,269
De-Es	672,813	13,246,255	14,362,615

Table 1. Training Corpus for European Languages

For the language models, we use the same data provided in the shared task. We also use the same development set and test set provided by the shared task. The in-domain test set includes 2,000 sentences and the out-of-domain test set includes 1,064 sentences for each language.

6.2 Translation System and Evaluation Method

To perform phrase-based SMT, we use Koehn's training scripts[1] and the Pharaoh decoder (Koehn, 2004). We run the decoder with its default settings and then use Koehn's implementation of minimum error rate training (Och, 2003) to tune the feature weights on the development set.

The translation quality was evaluated using a well-established automatic measure: BLEU score (Papineni et al., 2002). And we also use the tool provided in the NAACL/HLT 2006 shared task on SMT to calculate the BLEU scores.

6.3 Comparison of Different Lexical Weights

As described in section 4, we employ two methods to estimate the lexical weight in the translation model. In order to compare the two methods, we translate from French to Spanish, using English as the pivot language. We use the French-English and English-Spanish corpora described in Table 1 as training data. During training, before estimating the Spanish to French phrase translation probability, we filter those French-English and English-Spanish phrase pairs whose translation probabilities are below a fixed threshold 0.001.[2] The translation results are shown in Table 2.

[1] It is located at http://www.statmt.org/wmt06/shared-task/baseline.htm

[2] In the following experiments using pivot languages, we use the same filtering threshold for all of the language pairs.

The *phrase* method proposed in this paper performs better than the *word* method proposed in (Wang et al., 2006). This is because our method uses phrase translation probability as a confidence weight to estimate the lexical translation probability. It strengthens the frequently aligned pairs and weakens the infrequently aligned pairs. Thus, the following sections will use the *phrase* method to estimate the lexical weight.

Method	In-Domain	Out-of-Domain
Phrase	0.3212	0.2098
Word	0.2583	0.1672

Table 2. Results with Different Lexical Weights

6.4 Results of Using One Pivot Language

This section describes the translation results by using only one pivot language. For the language pair French and Spanish, we use English as the pivot language. The entire French-English and English-Spanish corpora as described in section 4 are used to train a pivot model for French-Spanish.

As described in section 5, if we have a small L_f-L_e bilingual corpus and large L_f-L_p and L_p-L_e bilingual corpora, we can obtain interpolated models.

In order to conduct the experiments, we randomly select 5K, 10K, 20K, 30K, 40K, 50K, and 100K sentence pairs from the French-Spanish corpus. Using each of these corpora, we train a standard translation model.

For each standard model, we interpolate it with the pivot model to get an interpolated model. The interpolation weights are tuned using the development set. For all the interpolated models, we set $\alpha_0 = 0.9$, $\alpha_1 = 0.1$, $\beta_0 = 0.9$, and $\beta_1 = 0.1$. We test the three kinds of models on both the in-domain and out-of-domain test sets. The results are shown in Figures 2 and 3.

The pivot model achieves BLEU scores of 0.3212 and 0.2098 on the in-domain and out-of-domain test set, respectively. It achieves an absolute improvement of 0.05 on both test sets (16.92% and 35.35% relative) over the standard model trained with 5,000 French-Spanish sentence pairs. And the performance of the pivot models are comparable with that of the standard models trained with 20,000 and 30,000 sentence pairs on the in-domain and out-of-domain test set, respectively. When the French-Spanish training corpus is increased, the standard models quickly outperform the pivot model.

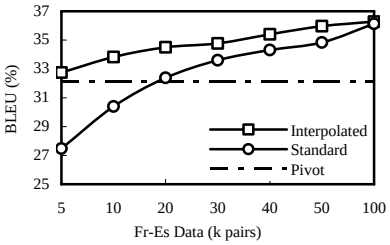


Figure 2. In-Domain French-Spanish Results

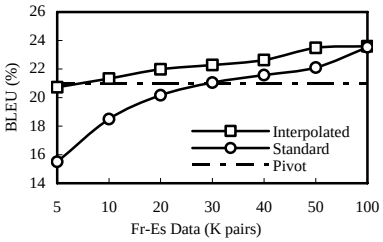


Figure 3. Out-of-Domain French-Spanish Results

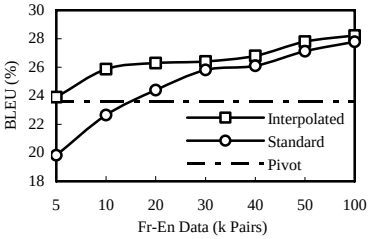


Figure 4. In-Domain French-English Results

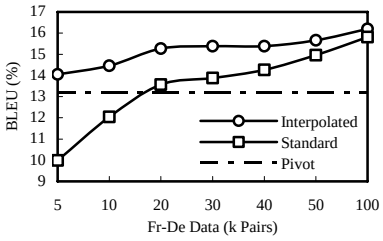


Figure 5. In-Domain French-German Results

When only a very small French-Spanish bilingual corpus is available, the interpolated method can greatly improve the translation quality. For example, when only 5,000 French-Spanish sentence pairs are available, the interpolated model outperforms the standard model by achieving a relative improvement of 17.55%, with the BLEU score improved from 0.2747 to 0.3229. With 50,000 French-Spanish sentence pairs available, the interpolated model significantly[3] improves the translation quality by achieving an absolute im-

[3] We conduct the significance test using the same method as described in (Koehn and Monz, 2006).

provement of 0.01 BLEU. When the French-Spanish training corpus increases to 100,000 sentence pairs, the interpolated model achieves almost the same result as the standard model. This indicates that our pivot language method is suitable for the language pairs with small quantities of training data available.

Besides experiments on French-Spanish translation, we also conduct translation from French to English and French to German, using German and English as the pivot language, respectively. The results on the in-domain test set[4] are shown in Figures 4 and 5. The tendency of the results is similar to that in Figure 2.

6.5 Results of Using More Than One Pivot Language

For French to Spanish translation, we also introduce German as a pivot language besides English. Using these two pivot languages, we build two different pivot models, and then perform linear interpolation on them. The interpolation weights for the English pivot model and the German pivot model are set to 0.6 and 0.4 respectively[5]. The translation results on the in-domain test set are 0.3212, 0.3077, and 0.3355 for the pivot models using English, German, and both German and English as pivot languages, respectively.

With the pivot model using both English and German as pivot languages, we interpolate it with the standard models trained with French-Spanish corpora of different sizes as described in the above section. The comparison of the translation results among the interpolated models, standard models, and the pivot model are shown in Figure 6.

It can be seen that the translation results can be further improved by using more than one pivot language. The pivot model "Pivot-En+De" using two pivot languages achieves an absolute improvement of 0.06 (22.13% relative) as compared with the standard model trained with 5,000 sentence pairs. And it achieves comparable translation result as compared with the standard model trained with 30,000 French-Spanish sentence pairs.

The results in Figure 6 also indicate the interpolated models using two pivot languages achieve the best results of all. Significance test shows that the interpolated models using two pivot languages significantly outperform those using one pivot language when less than 50,000 French-Spanish sentence pairs are available.

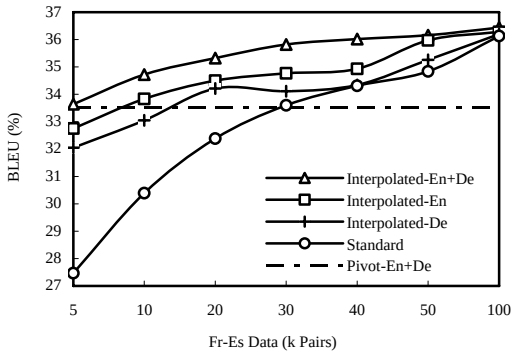


Figure 6. In-Domain French-Spanish Translation Results by Using Two Pivot Languages

6.6 Results by Using Pivot Language Related Corpora of Different Sizes

In all of the above results, the corpora used to train the pivot models are not changed. In order to examine the effect of the size of the pivot corpora, we decrease the French-English and English-French corpora. We randomly select 200,000 and 400,000 sentence pairs from both of them to train two pivot models, respectively. The translation results on the in-domain test set are 0.2376, 0.2954, and 0.3212 for the pivot models trained with 200,000, 400,000, and the entire French-English and English-Spanish corpora, respectively. The results of the interpolated models and the standard models are shown in Figure 7. The results indicate that the larger the training corpora used to train the pivot model are, the better the translation quality is.

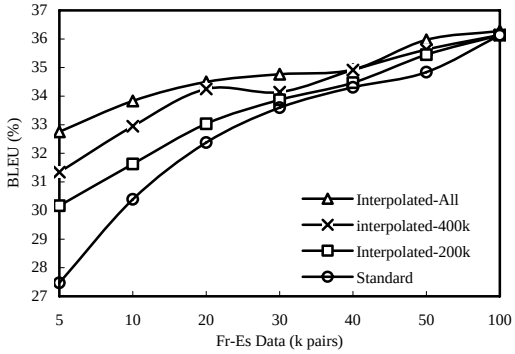


Figure 7. In-Domain French-Spanish Results by Using L_f-L_p and L_p-L_e Corpora of Different Sizes

[4] The results on the out-of-domain test set are similar to that in Figure 3. We only show the in-domain translation results in all of the following experiments because of space limit.

[5] The weights are tuned on the development set.

7 Experiments on Chinese to Japanese Translation

In section 6, translation results on the Europarl multilingual corpus indicate the effectiveness of our method. To investigate the effectiveness of our method by using independently sourced parallel corpora, we conduct Chinese-Japanese translation using English as a pivot language in this section, where the training data are not limited to a specific domain.

The data used for this experiment is the same as those used in (Wang et al., 2006). There are 21,977, 329,350, and 160,535 sentence pairs for the language pairs Chinese-Japanese, Chinese-English, and English-Japanese, respectively. The development data and testing data include 500 and 1,000 Chinese sentences respectively, with one reference for each sentence. For Japanese language model training, we use about 100M bytes Japanese corpus.

The translation result is shown in Figure 8. The pivot model only outperforms the standard model trained with 2,500 sentence pairs. This is because (1) the corpora used to train the pivot model are smaller as compared with the Europarl corpus; (2) the training data and the testing data are not limited to a specific domain; (3) The languages are not closely related.

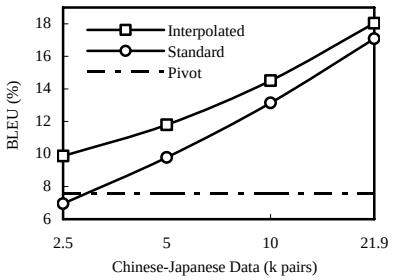


Figure 8. Chinese-Japanese Translation Results

The interpolated models significantly outperform the other models. When only 5,000 sentence pairs are available, the BLEU score increases relatively by 20.53%. With the entire (21,977 pairs) Chinese-Japanese available, the interpolated model relatively increases the BLEU score by 5.62%, from 0.1708 to 0.1804.

8 Conclusion

This paper proposed a novel method for phrase-based SMT on language pairs with a small bilingual corpus by bringing in pivot languages. To perform translation between L_f and L_e, we bring in a pivot language L_p, via which the large corpora of L_f-L_p and L_p-L_e can be used to induce a translation model for L_f-L_e. The advantage of this method is that it can perform translation between the language pair L_f-L_e even if no bilingual corpus for this pair is available. Using BLEU as a metric, our method achieves an absolute improvement of 0.06 (22.13% relative) as compared with the model directly trained with 5,000 sentence pairs for French-Spanish translation. And the translation quality is comparable with that of the model directly trained with 30,000 French-Spanish sentence pairs. The results also indicate that using more pivot languages leads to better translation quality.

With a small bilingual corpus available for L_f-L_e, we built a translation model, and interpolated it with the pivot model trained with the large L_f-L_p and L_p-L_e bilingual corpora. The results on both the Europarl corpus and Chinese-Japanese translation indicate that the interpolated models achieve the best results. Results also indicate that our pivot language approach is suitable for translation on language pairs with a small bilingual corpus. The less the L_f-L_e bilingual corpus is, the bigger the improvement is.

We also performed experiments using L_f-L_p and L_p-L_e corpora of different sizes. The results indicate that using larger training corpora to train the pivot model leads to better translation quality.

References

Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas. 2000. Learning Dependency Translation Models as Collections of Finite-State Head Transducers. *Computational Linguistics*, 26(1):45-60.

Niraj Aswani and Robert Gaizauskas. 2005. Aligning Words in English-Hindi Parallel Corpora. In *Proc. of the ACL 2005 Workshop on Building and Using Parallel Texts: Data-driven Machine Translation and Beyond*, pages 115-118.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2): 263-311.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved Statistical Machine Transla-

tion Using Paraphrases. In *Proc. of NAACL-2006*, pages 17-24.

Lars Borin. 2000. You'll Take the High Road and I'll Take the Low Road: Using a Third Language to Improve Bilingual Word Alignment. In *Proc. of COLING-2000*, pages 97-103.

David Chiang. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation. In *Proc. of ACL-2005*, pages 263-270.

Mona Diab and Philip Resnik. 2002. An Unsupervised Method for Word Sense Tagging using Parallel Corpora. In *Proc. of ACL-2002*, pages 255-262.

Tim Gollins and Mark Sanderson. 2001. Improving Cross Language Information Retrieval with Triangulated Translation. In *Proc. of ACM SIGIR-2001*, pages 90-95.

Kazuaki Kishida and Noriko Kando. 2003. Two-Stage Refinement of Query Translation in a Pivot Language Approach to Cross-Lingual Information Retrieval: An Experiment at CLEF 2003. In *Proc. of CLEF-2003*. pages 253-262.

Philipp Koehn. 2004. Pharaoh: A Beam Search Decoder for Phrase-Based Statistical Machine Translation Models. In *Proc. of AMTA-2004*, pages 115-124.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proc. of MT Summit X*, pages 79-86.

Philipp Koehn and Christof Monz. 2006. Manual and Automatic Evaluation of Machine Translation between European Languages. In *Proc. of the 2006 HLT-NAACL Workshop on Statistical Machine Translation*, pages 102-121.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proc. of HLT-NAAC- 2003*, pages 127-133.

Adam Lopez and Philip Resnik. 2005. Improved HMM Alignment Models for Languages with Scarce Resources. In *Proc. of the ACL-2005 Work-shop on Building and Using Parallel Texts: Data-driven Machine Translation and Beyond*, pages 83-86.

Joel Martin, Rada Mihalcea, and Ted Pedersen. 2005. Word Alignment for Languages with Scarce Resources. In *Proc. of the ACL-2005 Workshop on Building and Using Parallel Texts: Data-driven Machine Translation and Beyond*, pages 65-74.

Dan Melamed. 2004. Statistical Machine Translation by Parsing. In *Proc. of ACL-2004*, pages 653-660.

Bart Mellebeek, Karolina Owczarzak, Declan Groves, Josef Van Genabith, and Andy Way. 2006. A Syntactic Skeleton for Statistical Machine Translation. In *Proc. of EAMT-2006*, pages 195-202.

Sonja Niessen and Hermann Ney. 2004. Statistical Machine Translation with Scarce Resources Using Morpho-Syntactic Information. *Computational linguistics*, 30(2): 181-204.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proc. of ACL-2003*, pages 160-167.

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 *Proc. of ACL-2002*, pages 311-318.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency Treelet Translation: Syntactically Informed Phrasal SMT. In *Proc. of ACL-2005*, pages 271-279.

Tetsuya Sakai, Makoto Koyama, Akira Kumano, and Toshihiko Manabe. 2004. Toshiba BRIDJE at NTCIR-4 CLIR: Monolingual/Bilingual IR and Flexible Feedback. In *Proc. of NTCIR 4.*

Charles Schafer and David Yarowsky. 2002. Inducing Translation Lexicons via Diverse Similarity Measures and Bridge Languages. In *Proc. of CoNLL-2002*, pages 1-7.

Haifeng Wang, Hua Wu, and Zhanyi Liu. 2006. Word Alignment for Languages with Scarce Resources Using Bilingual Corpora of Other Language Pairs. In *Proc. of COLING/ACL-2006 Main Conference Poster Sessions*, pages 874-881.

Dan Tufis, Radu Ion, Alexandru Ceausu, and Dan Stefanescu. 2005. Combined Word Alignments. In *Proc. of the ACL-2005 Workshop on Building and Using Parallel Texts: Data-driven Machine Translation and Beyond*, pages 107-110.

Vincent Vandeghinste, Ineka Schuurman, Michael Carl, Stella Markantonatou, and Toni Badia. 2006. METIS-II: Machine Translation for Low-Resource Languages. In *Proc. of LREC-2006*, pages 1284-1289.

Dekai Wu. 1997. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, 23(3):377-403.

Kenji Yamada and Kevin Knight. 2001. A Syntax Based Statistical Translation Model. In *Proc. of ACL-2001*, pages 523-530.

Bootstrapping a Stochastic Transducer for Arabic-English Transliteration Extraction

Tarek Sherif and **Grzegorz Kondrak**
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2E8
{tarek,kondrak}@cs.ualberta.ca

Abstract

We propose a bootstrapping approach to training a memoriless stochastic transducer for the task of extracting transliterations from an English-Arabic bitext. The transducer learns its similarity metric from the data in the bitext, and thus can function directly on strings written in different writing scripts without any additional language knowledge. We show that this bootstrapped transducer performs as well or better than a model designed specifically to detect Arabic-English transliterations.

1 Introduction

Transliterations are words that are converted from one writing script to another on the basis of their pronunciation, rather than being translated on the basis of their meaning. Transliterations include named entities (e.g. جين اوستن/*Jane Austen*) and lexical loans (e.g. تلفيزيون/*television*).

An algorithm to detect transliterations automatically in a bitext can be an effective tool for many tasks. Models of machine transliteration such as those presented in (Al-Onaizan and Knight, 2002) or (AbdulJaleel and Larkey, 2003) require a large set of sample transliterations to use for training. If such a training set is unavailable for a particular language pair, a detection algorithm would lead to a significant gain in time over attempting to build the set manually. Algorithms for cross-language information retrieval often encounter the problem of out-of-vocabulary words, or words not present in the algorithm's lexicon. Often, a significant proportion of these words are named entities and thus are candidates for transliteration. A transliteration detection algorithm could be used to map named entities in a query to potential transliterations in the target language text.

The main challenge in transliteration detection lies in the fact that transliteration is a *lossy* process. In other words, information can be lost about the original word when it is transliterated. This can occur because of phonetic gaps in one language or the other. For example, the English [p] sound does not exist in Arabic, and the Arabic [ʕ] sound (made by the letter ع) does not exist in English. Thus, *Paul* is transliterated as بول [bul], and علي [ʕali] is transliterated as *Ali*. Another form of löss occurs when the relationship between the orthographic and phonetic representations of a word are unclear. For example, the [k] sound will always be written with the letter ك in Arabic, but in English it can be written as *c*, *k ch*, *ck*, *cc* or *kk* (not to mention being one of the sounds produced by *x*). Finally, letters may be deleted in one language or the other. In Arabic, short vowels will often be omitted (e.g. يوسف/*Yousef*), while in English the Arabic ء and ع are often deleted (e.g. اسماعيل/*Ismael*).

We explore the use of word similarity metrics on the task of Arabic-English transliteration detection and extraction. One of our primary goals in exploring these metrics is to assess whether it is possible maintain high performance without making the algorithms language-specific. Many word-similarity metrics require that the strings being compared be

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 864–871,
Prague, Czech Republic, June 2007.

written in the same script. Levenshtein edit distance, for example, does not produce a meaningful score in the absence of character identities. Thus, if these metrics are to be used for transliteration extraction, modifications must be made to allow them to compare different scripts.

Freeman et al. (2006) take the approach of manually encoding a great deal of language knowledge directly into their Arabic-English fuzzy matching algorithm. They define equivalence classes between letters in the two scripts and perform several rule-based transformations to make word pairs more comparable. This approach is unattractive for two reasons. Firstly, predicting all possible relationships between letters in English and Arabic is difficult. For example, allowances have to be made for unusual pronunciations in foreign words such as the *ch* in *cliché* or the *c* in *Milosevic*. Secondly, the algorithm becomes completely language-specific, which means that it cannot be used for any other language pair.

We propose a method to learn letter relationships directly from the bitext containing the transliterations. Our model is based on the memoriless stochastic transducer proposed by Ristad and Yianilos (1998), which derives a probabilistic word-similarity function from a set of examples. The transducer is able to learn edit distance costs between disjoint sets of characters representing different writing scripts without any language-specific knowledge. The transducer approach, however, requires a large set of training examples, which is a limitation not present in the fuzzy matching algorithm. Thus, we propose a bootstrapping approach (Yarowsky, 1995) to train the stochastic transducer iteratively as it extracts transliterations from a bitext. The bootstrapped stochastic transducer is completely language-independent, and we show that it is able to perform at least as well as the Arabic-English specific fuzzy matching algorithm.

The remainder of this paper is organized as follows. Section 2 presents our bootstrapping method to train a stochastic transducer. Section 3 outlines the Arabic-English fuzzy matching algorithm. Section 4 discusses other word-similarity models used for comparison. Section 5 describes the results of two experiments performed to test the models. Section 6 briefly discusses previous approaches to detecting transliterations. Section 7 presents our conclusions and possibilities for future work.

2 Bootstrapping with a Stochastic Transducer

Ristad and Yianilos (1998) propose a probabilistic framework for word similarity, in which the similarity of a pair of words is defined as the sum of the probabilities of all paths through a memoriless stochastic transducer that generate the pair of words. This is referred to as the forward score of the pair of words. They outline a forward-backward algorithm to train the model and show that it outperforms Levenshtein edit distance on the task of pronunciation classification.

The training algorithm begins by calling the forward (Equation 1) and backward (Equation 2) functions to fill in the F and B tables for training pair s and t with respective lengths I and J.

$$\begin{aligned} F(0,0) &= 1 \\ F(i,j) &= P(s_i,\epsilon)F(i-1,j) \\ &\quad +P(\epsilon,t_j)F(i,j-1) \\ &\quad +P(s_i,t_j)F(i-1,j-1) \end{aligned} \tag{1}$$

$$\begin{aligned} B(I,J) &= 1 \\ B(i,j) &= P(s_{i+1},\epsilon)B(i+1,j) \\ &\quad +P(\epsilon,t_{j+1})B(i,j+1) \\ &\quad +P(s_{i+1},t_{j+1})B(i+1,j+1) \end{aligned} \tag{2}$$

The forward value at each position (i,j) in the F matrix signifies the sum of the probabilities of all paths through the transducer that produce the prefix pair (s_1^i, t_1^j), while $B(i,j)$ contains the sum of the probabilities of all paths through the transducer that generate the suffix pair (s_{i+1}^I, t_{j+1}^J). These tables can then be used to collect partial counts to update the probabilities. For example, the mapping (s_i, t_j) would contribute a count according to Equation 3. These counts are then normalized to produce the updated probability distribution.

$$C(s_i,t_j) += \frac{F(i-1,j-1)P(s_i,t_j)B(i,j)}{F(I,J)} \tag{3}$$

The major issue in porting the memoriless transducer over to our task of transliteration extraction

is that its training is supervised. In other words, it would require a relatively large set of known transliterations for training, and this is exactly what we would like the model to acquire. In order to overcome this problem, we look to the bootstrapping method outlined in (Yarowsky, 1995). Yarowsky trains a rule-based classifier for word sense disambiguation by starting with a small set of seed examples for which the sense is known. The trained classifier is then used to label examples for which the sense is unknown, and these newly labeled examples are then used to retrain the classifier. The process is repeated until convergence.

Our method uses a similar approach to train the stochastic transducer. The algorithm proceeds as follows:

1. Initialize the training set with the seed pairs.
2. Train the transducer using the forward-backward algorithm on the current training set.
3. Calculate the forward score for all word pairs under consideration.
4. If the forward score for a pair of words is above a predetermined acceptance threshold, add the pair to the training set.
5. Repeat steps 2-4 until the training set ceases to grow.

Once training stops, the transducer can be used to score pairs of words not in the training set. For our experiments, the acceptance threshold was optimized on a separate development set. Forward scores were normalized by the average of the lengths of the two words.

3 Arabic-English Fuzzy String Matching

In this section, we outline the fuzzy string matching algorithm proposed by Freeman et al. (2006). The algorithm is based on the standard Levenshtein distance approach, but encodes a great deal of knowledge about the relationships between English and Arabic letters.

Initially, the candidate word pair is modified in two ways. The first transformation is a rule-based letter normalization of both words. Some examples of normalization include:

- English double letter collapse: e.g. *Miller→Miler*.

ا,أ,آ,ى ↔ *a,e,i,o,u*	ب ↔ *b,p,v*
ت,ط,ث ↔ *t*	ج ↔ *j,g*
ظ ↔ *d,z*	ع,ء ↔ *',c,a,e,i,o,u*
ق ↔ *q,g,k*	ك ↔ *k,c,s*
ي ↔ *y,i,e,j*	ة ↔ *a,e*

Table 1: A sample of the letter equivalence classes for fuzzy string matching.

Algorithm VowelNorm $(Estring, Astring)$

for each $i := 0$ **to** $min(|Estring|, |Astring|)$
  for each $j := 0$ **to** $min(|Estring|, |Astring|)$
    if $Astring_i = Estring_j$
      $Outstring. = Estring_j; i++; j++;$
    if $vowel(Astring_i) \wedge vowel(Estring_j)$
      $Outstring. = Estring_j; i++; j++;$
    if $\neg vowel(Astring_i) \wedge vowel(Estring_j)$
      $j++;$
      if $j < |Estring_j|$
        $Outstring. = Estring_j; i++; j++;$
    else
      $Outstring. = Estring_j; i++; j++;$
while $j < |Estring|$
  if $\neg vowel(Estring_j)$
    $Outstring. = Estring_j;$
  $j++;$
return $Outstring;$

Figure 1: Pseudocode for the vowel transformation procedure.

- Arabic hamza collapse: e.g. أشرف→اشرف.
- Individual letter normalizations: e.g. *Hendrix→Hendriks* or شريف→سهريف.

The second transformation is an iteration through both words to remove any vowels in the English word for which there is no similarly positioned vowel in the Arabic word. The pseudocode for our implementation of this vowel transformation is presented in Figure 1.

After letter and vowel transformations, the Levenshtein distance is computed using the letter equivalences as matches instead of identities. Some equivalence classes between English and Arabic letters are shown in Table 1. The Arabic and English letters within a class are treated as identities. For example, the Arabic ف can match both *f* and *v* in English with no cost. The resulting Levenshtein distance is normalized by the sum of the lengths of both words.

	Levenshtein	ALINE	Fuzzy Match	Bootstrap
Lang.-specific	No	No	Yes	No
Preprocessing	Romanization	Phon. Conversion	None	None
Data-driven	No	No	No	Yes

Table 2: Comparison of the word-similarity models.

Several other modifications, such as light stemming and multiple passes to discover more difficult mappings, were also proposed, but they were found to influence performance minimally. Thus, the equivalence classes and transformations are the only modifications we reproduce for our experiments here.

4 Other Models of Word Similarity

In this section, we present two models of word similarity used for purposes of comparison. Levenshtein distance and ALINE are not language-specific per se, but require that the words being compared be written in a common script. Thus, they require some language knowledge in order to convert one or both of the words into the common script. A comparison of all the models presented is given in Table 2.

4.1 Levenshtein Edit Distance

As a baseline for our experiments, we used Levenshtein edit distance. The algorithm simply counts the minimum number of insertions, deletions and substitutions required to convert one string into another. Levenshtein distance depends on finding identical letters, so both words must use the same alphabet. Prior to comparison, we convert the Arabic words into the Latin alphabet using the intuitive mappings for each letter shown in Table 3. The distances are also normalized by the length of the longer of the two words to avoid excessively penalizing longer words.

4.2 ALINE

Unlike other algorithms presented here, the ALINE algorithm (Kondrak, 2000) operates in the phonetic, rather than the orthographic, domain. It was originally designed to identify cognates in related languages, but it can be used to compute similarity between any pair of words, provided that they are expressed in a standard phonetic notation. Individual

ا,أ,آ,ء → *a*	ب → *b*	ت,ط → *t*
ة → *a*	ث → *th*	ج → *j*
ه,ح → *h*	خ → *kh*	د,ض → *d*
ذ,ظ → *th*	ر → *r*	ز → *z*
س,ص → *s*	ش → *sh*	ع → ’
غ → *g*	ف → *f*	ق → *q*
ك → *k*	ل → *l*	م → *m*
ن → *n*	و → *w*	ي → *y*

Table 3: Arabic Romanization for Levenshtein distance.

phonemes input to the algorithm are decomposed into a dozen phonetic features, such as Place, Manner and Voice. A substitution score between a pair of phonemes is based on the similarity as assessed by a comparison of the individual features. After an optimal alignment of the two words is computed with a dynamic programming algorithm, the overall similarity score is set to the sum of the scores of all links in the alignment normalized by the length of the longer of the two words.

In our experiments, the Arabic and English words were converted into phonetic transcriptions using a deterministic rule-based transformation. The transcriptions were only approximate, especially for English vowels. Arabic emphatic consonants were depharyngealized.

5 Evaluation

The word-similarity metrics were evaluated on two separate tasks. In experiment 1 (Section 5.1) the task was to extract transliterations from a sentence aligned bitext. Experiment 2 (Section 5.2) provides the algorithms with named entities from an English document and requires them to extract the transliterations from the document’s Arabic translation.

The two bitexts used in the experiments were the

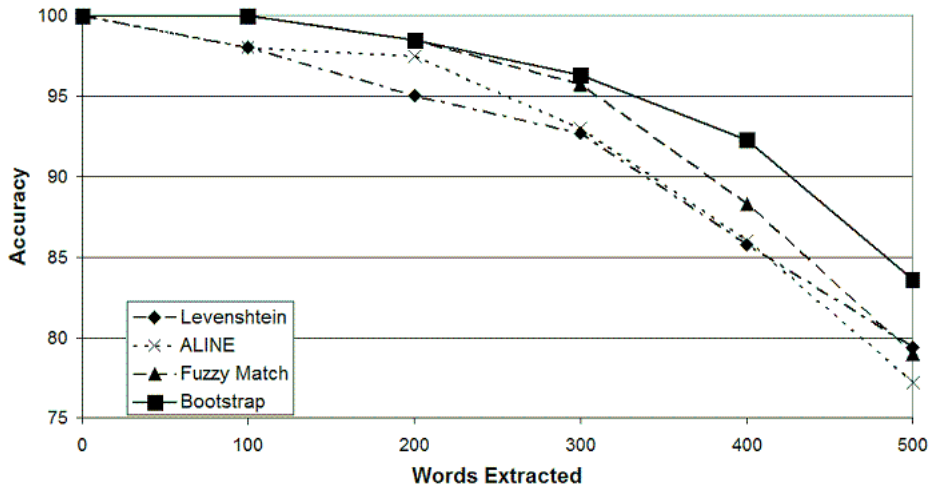


Figure 2: Precision per number of words extracted for the various algorithms from a sentence-aligned bitext.

Arabic Treebank Part 1-10k word English Translation corpus and the Arabic English Parallel News Part 1 corpus (approx. 2.5M words). Both bitexts contain Arabic news articles and their English translations aligned at the sentence level, and both are available from the Linguistic Date Consortium. The Treebank data was used as a development set to optimize the acceptance threshold used by the bootstrapped transducer. Testing for the sentence-aligned extraction task was done on the first 20k sentences (approx. 50k words) of the parallel news data, while the named entity extraction task was performed on the first 1000 documents of the parallel news data. The seed set for bootstrapping the stochastic transducer was manually constructed and consisted of 14 names and their transliterations.

5.1 Experiment 1: Sentence-Aligned Data

The first task used to test the models was to compare and score the words remaining in each bitext sentence pair after preprocessing the bitext in the following way:

- The English corpus is tokenized using a modified[1] version of Word Splitter[2].
- All uncapitalized English words are removed.
- Stop words (mainly prepositions and auxiliary verbs) are removed from both sides of the bitext.
- Any English words of length less than 4 and Arabic words of length less than 3 are removed.

Each algorithm finds the top match for each English word and the top match for each Arabic word. If two words mark each other as their top scorers, then the pair is marked as a transliteration pair. This one-to-one constraint is meant to boost precision, though it will also lower recall. This is because for many of the tasks in which transliteration extraction would be useful (such as building a lexicon), precision is deemed more important. Transliteration pairs are sorted according to their scores, and the top 500 hundred scoring pairs are returned.

The results for the sentence-aligned extraction task are presented in Figure 2. Since the number of actual transliterations in the data was unknown, there was no way to compute recall. The measure used here is the precision for each 100 words extracted up to 500. The bootstrapping method is equal to or outperforms the other methods at all levels, including the Arabic-English specific fuzzy match algorithm. Fuzzy matching does well for the first few hundred words extracted, but eventually falls below the level of the baseline Levenshtein.

Interestingly, the bootstrapped transducer does not seem to have trouble with digraphs, despite the one-to-one nature of the character operations. Word pairs with two-to-one mappings such as *sh*/ش or

[1]The way the program handles apostrophes(') had to be modified since they are sometimes used to represent glottal stops in transliterations of Arabic words, e.g. qala'a.

[2]Available at http://l2r.cs.uiuc.edu/~cogcomp/tools.php.

	Metric	Arabic	Romanized	English
1	Bootstrap	الاخيرين	alakhyryn	Algerian
2	Bootstrap	وسلم	wslm	Islam
3	Fuzzy M.	لكل	lkl	Alkella
4	Fuzzy M.	عمان	'mAn	common
5	ALINE	سكر	skr	sugar
6	Leven.	أصاب	asab	Arab
7	All	مارك	mark	Marks
8	All	روسيون	rwsywn	Russian
9	All	استراتيجية	istratyjya	strategic
10	All	فرنك	frnk	French

Table 4: A sample of the errors made by the word-similarity metrics.

x/كس tend to score lower than their counterparts composed of only one-to-one mappings, but nevertheless score highly.

A sample of the errors made by each word-similarity metric is presented in Table 4. Errors 1-6 are indicative of the weaknesses of each individual algorithm. The bootstrapping method encounters problems when erroneous pairs become part of the training data, thereby reinforcing the errors. The only problematic mapping in Error 1 is the خ/*g* mapping, and thus the pair has little trouble getting into the training data. Once the pair is part of training data, the algorithm learns that the mapping is acceptable and uses it to acquire other training pairs that contain the same erroneous mapping. The problem with the fuzzy matching algorithm seems to be that it creates too large a class of equivalent words. The pairs in errors 3 and 4 are given a total edit cost of 0. This is possible because of the overly general letter and vowel transformations, as well as unusual choices made for letter equivalences (e.g. ع/*c* in error 4). ALINE's errors tend to occur when it links two letters, based on phonetic similarity, that are never mapped to each other in transliteration because they each have a more direct equivalent in the other language (error 5). Although the Arabic ك [k] is phonetically similar to the English *g*, they would never be mapped to each other since English has several ways of representing an actual [k] sound. Errors made by Levenshtein distance (error 6) are simply due to the fact that it considers all non-identity mappings to be equivalent.

Errors 7-10 are examples of general errors made by all the algorithms. The most common error was related to inflection (error 7). The words are essentially transliterations of each other, but one or the other of the two words takes a plural or some other inflectional ending that corrupts the phonetic match. Error 8 represents the common problem of incidental letter similarity. The English *-ian* ending used for nationalities is very similar to the Arabic يون [ijun] and يين [ijin] endings which are used for the same purpose. They are similar phonetically and, since they are functionally similar, will tend to co-occur. Since neither can be said to be derived from the other, however, they cannot be considered transliterations. Error 9 is a case of two words of common origin taking on language-specific derivational endings that corrupt the phonetic match. Finally, error 10 shows a mapping (ك/*c*) that is often correct in transliteration, but is inappropriate in this particular case.

5.2 Experiment 2: Document-Aligned Named Entity Recognition

The second experiment provides a more challenging task for the evaluation of the models. It is structured as a cross-language named entity recognition task similar to those outlined in (Lee and Chang, 2003) and (Klementiev and Roth, 2006). Essentially, the goal is to use a language for which named entity recognition software is readily available as a reference for tagging named entities in a language for which such software is not available. For this task, the sentence alignment of the bitext is ignored. For each named entity in an English document, the models must select a transliteration from within the document's entire Arabic translation. This is meant to be a loose approximation of the "comparable" corpora used in (Klementiev and Roth, 2006). The comparable corpora are related documents in different languages that are not translations (e.g. news articles describing the same event), and thus sentence alignment is not possible.

The first 1000 documents in the parallel news data were used for testing. The English side of the bitext was tagged with Named Entity Tagger[3], which labels named entities as *person*, *location*, *organiza-*

[3]Available at http://l2r.cs.uiuc.edu/~cogcomp/tools.php.

Method	Accuracy
Levenshtein	69.3
ALINE	71.9
Fuzzy Match	74.6
Bootstrapping	74.6

Table 5: Precision of the various algorithms on the NER detection task.

	Metric	Arabic	Romanized	English
1	Both	عبد	’bd	Abdallah
2	Bootstrap	العديد	al’dyd	Alhadidi
3	Fuzzy Match	ثمن	thmn	Othman

Table 6: A sample of errors made on the NER detection task.

tion or *miscellaneous*. The words labeled as *person* were extracted. Person names are almost always transliterated, while for the other categories this is far less certain. The list was then hand-checked to ensure that all names were candidates for transliteration, leaving 822 names. The restrictions on word length and stop words were the same as before, but in this task each of the English person names from a given document were compared to all valid words in the corresponding Arabic document, and the top scorer for each English name was returned.

The results for the NER detection task are presented in Table 5. It seems the bootstrapped transducer's advantage is relative to the proportion of correct transliteration pairs to the total number of candidates. As this proportion becomes smaller the transducer is given more opportunities to corrupt its training data and performance is affected accordingly. Nevertheless, the transducer is able to perform as well as the language-specific fuzzy matching algorithm on this task, despite the greater challenge posed by selecting candidates from entire documents.

A sample of errors made by the bootstrapped transducer and fuzzy matching algorithms is shown in Table 6. Error 1 was due to the fact that names are sometimes split differently in Arabic and English. The Arabic عبد الله (2 words) is generally written as *Abdallah* in English, leading to partial matches with part of the Arabic name. Error 2 shows an issue with the one-to-one nature of the transducer. The deleted *h* can be learned in mappings such as *sh*/ش or *ph*/ف, but it is generally inappropriate to delete an *h* on its own. Error 3 again shows that the fuzzy matching algorithm's letter transformations are too general. The vowel removals lead to a 0 cost match in this case.

6 Related Work

Several other methods for detecting transliterations between various language pairs have been proposed. These methods differ in their complexity as well as in their applicability to language pairs other than the pair for which they were originally designed.

Collier et al. (1997) present a method for identifying transliterations in an English-Japanese bitext. Their model first transcribes the Japanese word expressed in the *katakana* syllabic script as the concatenation of all possible transliterations of the individual symbols. A depth-first search is then applied to compute the number of matches between this transcription and a candidate English transliteration. The method requires a manual enumeration of the possible transliterations for each katakana symbol, which is unfeasible for many language pairs.

In the method developed by Tsuji (2002), katakana strings are first split into their mora units, and then the transliterations of the units are assessed manually from a set of training pairs. For each katakana string in a bitext, all possible transliterations are produced based on the transliteration units determined from the training set. The transliteration candidates are then compared to the English words according to the Dice score. The manual enumeration of possible mappings makes this approach unattractive for many language pairs, and the generation of all possible transliteration candidates is problematic in terms of computational complexity.

Lee and Chang (2003) detect transliterations with a generative noisy channel transliteration model similar to the transducer presented in (Knight and Graehl, 1998). The English side of the corpus is tagged with a named entity tagger, and the model is used to isolate the transliterations in the Chinese translation. This model, like the transducer proposed by Ristad and Yianilos (1998), must be trained on a large number of sample transliterations, meaning it cannot be used if such a resource is not avail-

able.

Klementiev and Roth (2006) bootstrap with a perceptron and use temporal analysis to detect transliterations in comparable Russian-English news corpora. The English side is first tagged by a named entity tagger, and the perceptron proposes transliterations for the named entities. The candidate transliteration pairs are then reranked according the similarity of their distributions across dates, as calculated by a discrete Fourier transform.

7 Conclusion and Future Work

We presented a bootstrapping approach to training a stochastic transducer, which learns scoring parameters automatically from a bitext. The approach is completely language-independent, and was shown to perform as well or better than an Arabic-English specific similarity metric on the task of Arabic-English transliteration extraction.

Although the bootstrapped transducer is language-independent, it learns only one-to-one letter relationships, which is a potential drawback in terms of porting it to other languages. Our model is able to capture English digraphs and trigraphs, but, as of yet, we cannot guarantee the model's success on languages with more complex letter relationships (e.g. a logographic writing system such as Chinese). More research is necessary to evaluate the model's performance on other languages.

Another area open to future research is the use of more complex transducers for word comparison. For example, Linden (2006) presents a model which learns probabilities for edit operations by taking into account the context in which the characters appear. It remains to be seen how such a model could be adapted to a bootstrapping setting.

Acknowledgments

We would like to thank the members of the NLP research group at the University of Alberta for their helpful comments and suggestions. This research was supported by the Natural Sciences and Engineering Research Council of Canada.

References

N. AbdulJaleel and L. S. Larkey. 2003. Statistical transliteration for English-Arabic cross language information retrieval. In *CIKM*, pages 139–146.

Y. Al-Onaizan and K. Knight. 2002. Machine transliteration of names in Arabic text. In *ACL Workshop on Comp. Approaches to Semitic Languages*.

N. Collier, A. Kumano, and H. Hirakawa. 1997. Acquisition of English-Japanese proper nouns from noisy-parallel newswire articles using Katakana matching. In *Natural Language Pacific Rim Symposium (NLPRS'97), Phuket, Thailand*, pages 309–314, December.

A. Freeman, S. Condon, and C. Ackerman. 2006. Cross linguistic name matching in English and Arabic. In *Human Language Technology Conference of the NAACL*, pages 471–478, New York City, USA, June. Association for Computational Linguistics.

A. Klementiev and D. Roth. 2006. Named entity transliteration and discovery from multilingual comparable corpora. In *Human Language Technology Conference of the NAACL*, pages 82–88, New York City, USA, June. Association for Computational Linguistics.

K. Knight and J. Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599–612.

G. Kondrak. 2000. A new algorithm for the alignment of phonetic sequences. In *NAACL 2000*, pages 288–295.

C. Lee and J. S. Chang. 2003. Acquisition of English-Chinese transliterated word pairs from parallel-aligned texts using a statistical machine transliteration model. In *HLT-NAACL 2003 Workshop on Building and using parallel texts*, pages 96–103, Morristown, NJ, USA. Association for Computational Linguistics.

K. Linden. 2006. Multilingual modeling of cross-lingual spelling variants. *Information Retrieval*, 9(3):295–310, June.

E. S. Ristad and P. N. Yianilos. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532.

K. Tsuji. 2002. Automatic extraction of translational Japanese-katakana and English word pairs. *International Journal of Computer Processing of Oriental Languages*, 15(3):261–279.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Meeting of the Association for Computational Linguistics*, pages 189–196.

Benefits of the 'Massively Parallel Rosetta Stone': Cross-Language Information Retrieval with over 30 Languages

Peter A. Chew
Sandia National Laboratories
P. O. Box 5800, MS 1012
Albuquerque, NM 87185-1012, USA
pchew@sandia.gov

Ahmed Abdelali
New Mexico State University
P.O. Box 30002, Mail Stop 3CRL
Las Cruces, NM 88003-8001, USA
ahmed@crl.nmsu.edu

Abstract

In this paper, we describe our experiences in extending a standard cross-language information retrieval (CLIR) approach which uses parallel aligned corpora and Latent Semantic Indexing. Most, if not all, previous work which follows this approach has focused on bilingual retrieval; two examples involve the use of French-English or English-Greek parallel corpora. Our extension to the approach is 'massively parallel' in two senses, one linguistic and the other computational. First, we make use of a parallel aligned corpus consisting of almost 50 parallel translations in over 30 distinct languages, each in over 30,000 documents. Given the size of this dataset, a 'massively parallel' approach was also necessitated in the more usual computational sense. Our results indicate that, far from adding more noise, more linguistic parallelism is better when it comes to cross-language retrieval precision, in addition to the self-evident benefit that CLIR can be performed on more languages.

1 Introduction

Approaches to cross-language information retrieval (CLIR) fall generally into one of two types, or some combination thereof: the 'query translation' approach or the 'parallel corpus' approach. The first of these, which is perhaps more common, involves translation of the query into the target language, for example using machine translation or on-line dictionaries. The second makes use of parallel aligned corpora as training sets. One approach which uses parallel corpora does this in conjunction with Latent Semantic Indexing (LSI) (Landauer and Littman 1990, Young 1994). According to Berry et al. (1994:21), the use of LSI with parallel corpora can be just as effective as the query translation approach, and avoids some of the drawbacks of the latter, discussed in Nie et al. (1999).

Generally, research in CLIR has not attempted to use very many languages at a time (see for example Nie and Jin 2002). With query translation (although that is not the approach that Nie and Jin take), this is perhaps understandable, as for each new language, a new translation algorithm must be included. The effort involved in extending query translation to multiple languages, therefore, is likely to be in proportion to the number of languages.

With parallel corpora, the reason that research has been limited to only a few languages at a time – and usually just two at a time, as in the LSI work cited above – is more likely to be rooted in the widespread perception that good parallel corpora are difficult to obtain (see for example Asker 2004). However, recent work (Resnik et al. 1999, Chew et al. 2006) has challenged this idea.

One advantage of a 'massively parallel' multilingual corpus is perhaps self-evident: within the LSI framework, the more languages are mapped into the single conceptual space, the fewer restrictions there are on which languages documents can be selected from for cross-language retrieval. However, several questions were raised for us as

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 872–879,
Prague, Czech Republic, June 2007.

we contemplated the use of a massively parallel corpus. Would the addition of languages not used in testing create 'noise' for a given language pair, reducing the precision of CLIR? Could *partially* parallel corpora be used? Our work appears to show both that more languages are generally beneficial, and even incomplete parallel corpora can be used. In the remainder of this paper, we provide evidence for this claim. The paper is organized as follows: section 2 describes the work we undertook to build the parallel corpus and its characteristics. In section 3, we outline the mechanics behind the 'Rosetta-Stone' type method we use for cross-language comparison. In section 4, we present and discuss the results of the various tests we performed. Finally, we conclude on our findings in section 5.

2 The massively parallel corpus

Following Chew et al. (2006), our parallel corpus was built up from translations of the Bible which are freely available on the World Wide Web. Although reliable comparable statistics are hard to find, it appears to be generally agreed that the Bible is the world's most widely translated book, with complete translations in 426 languages and partial translations in 2,403 as of December 31, 2005 (Bible Society, 2006). Great care is taken over the translations, and they are alignable by chapter and verse. According to Resnik et al. (1999), the Bible's coverage of modern vocabulary may be as high as 85%. The vast majority of the translations we used came from the 'Unbound Bible' website (Biola University, 2005-2006); from this website, the text of a large number of different translations of the Bible can – most importantly for our purposes – be downloaded in a tab-delimited format convenient for loading into a database and then indexing by chapter and verse in order to ensure 'parallelism' in the corpus. The number of translations available at the website is apparently being added to, based on our observations accessing the website on a number of different occasions.

The languages we have included in our multilingual parallel corpus include those both ancient and modern, and are as follows:

Language	No. of translations	Used in tests
Afrikaans	1	12+
Albanian	1	27+
Arabic	1	All
Chinese (Simplified)	1	44+
Chinese (Traditional)	1	44+
Croatian	1	27+
Czech	2	12+
Danish	1	12+
Dutch	1	12+
English	7	All
Finnish	3	27+
French	2	All
German	4	8,27+
Greek (New Testament)	2	46+
Hebrew (Old Testament)	1	46+
Hebrew (Modern)	1	6,12+
Hungarian	1	6+
Italian	2	8,27+
Japanese*	1	9+
Korean	1	27+
Latin	1	8,9,28+
Maori	1	7,8,9,27+
Norwegian	1	27+
Polish*	1	27+
Portuguese	1	27+
Russian	1	All
Spanish	2	All
Swedish	1	27+
Tagalog	1	27+
Thai	1	27+
Vietnamese	1	27,44+

Table 1. Languages[1]

The languages above represent many of the major language groups: Austronesian (Maori and Tagalog); Altaic (Japanese and Korean); Sino-Tibetan (Chinese); Semitic (Arabic and Hebrew); Finno-Ugric (Finnish and Hungarian); Austro-Asiatic (Vietnamese); Tai-Kadai (Thai); and Indo-European (the remaining languages). The two New Testament Greek versions are the Byzantine/Majority Text (2000), and the parsed version of the same text, in which we treated distinct morphological elements (such as roots or inflectional endings) as distinct terms. Overall, the list includes

[1] Translations in languages marked with an asterisk above were obtained from websites other than the 'Unbound Bible' website. 'Used in tests' indicates in which tests in Table 2 below the language was used as training data, and hence the order of addition of languages to the training data.

47 versions in 31 distinct languages (assuming without further discussion here that each entry in the list represents a distinct language).

We aligned the translations by verse, and, since there are some differences in versification between translations (for example, the Hebrew Old Testament includes the headings for the Psalms as separate verses, unlike most translations), we spent some time cleaning the data to ensure the alignment was as good as possible, given available resources and our knowledge of the languages. (Even after this process, the alignment was not perfect, and differences in how well the various translations were aligned may account for some of the variability in the outcome of our experiments, depending on which translations were used.) The end result was that our parallel corpus consisted of 31,226 'mini-documents' – the total number of text chunks[2] after the cleaning process, aligned across all 47 versions. The two New Testament Greek versions, and the one Old Testament Hebrew version, were exceptions because these are only partially complete; the former have text in only 7,953 of the verses, and the latter has text in 23,266 of the verses. For some versions, a few of the verse translations are incomplete where a particular verse has been skipped in translation; this also explains the fact that the number of Hebrew and Greek text chunks together do not add up to 31,226. However, the number of such verses is negligible in comparison to the total.

3 Framework

The framework we used was the standard LSI framework described in Berry et al. (1994). Each aligned mini-document from the parallel corpus consists of the *combination* of text from all the 31 languages. A document-by-term matrix is formed in which each cell represents a weighted frequency of a particular term *t* in a particular document *k*. We used a standard log-entropy weighting scheme, where the weighted frequency W is given by:

$$W = \log_2 (F) \times (1 + H_t / \log_2 (N))$$

where F is the raw frequency of *t* in *k*, H_t is the standard '*p* log *p*' measure of the entropy of the term across all documents, and N is the number of documents in the corpus. The last term in the expression above, $\log_2 (N)$, is the maximum entropy that any term can have in the corpus, and therefore $(1 + H_t / \log_2 (N))$ is 1 for the most distinctive terms in the corpus, 0 for those which are least distinctive.

The sparse document-by-term matrix is subjected to singular value decomposition (SVD), and a reduced non-sparse matrix is output. Generally, we used the output corresponding to the top 300 singular values in our experiments. When we had a smaller number of languages in the mix, it was possible to use SVDPACK (Berry et al. 1996), which is an open-source non-parallel algorithm for computing the SVD, but for larger problems (involving more than a couple of dozen parallel versions), use of a parallel algorithm (in a library called Trilinos) was necessitated. (This was run on a Linux cluster consisting of 4,096 dual CPU compute nodes, running on Dell PowerEdge 1850 1U Servers with 6GB of RAM.)

In order to test the precision versus recall of our framework, we used translations of the 114 suras of the Qu'ran into five languages, Arabic, English, French, Russian and Spanish. The number of documents used for testing is fairly small, but large enough to give comparative results for our purposes which are still highly statistically significant. The test set was split into each of the 10 possible language-pair combinations: Arabic-English, Arabic-French, English-French, and so on.

For each language pair and test, 228 distinct 'queries' were submitted – each query consisting of one of the 228 sura 'documents'. If the highest-ranking document in the other language of the pair was in fact the query's translation, then the result was deemed 'correct'. To assess the aggregate performance of the framework, we used two measures: average precision at 0 (the maximum precision at any level of recall), and average precision at 1 document (1 if the 'correct' document ranked highest, zero otherwise). The second measure is a stricter one, but we generally found that there is a high rate of correlation between the two measures anyway.

4 Results and Discussion

The following tables show the results of our tests. First, we present in Table 2 the overall summary,

[2] The text chunks generally had the same boundaries as the verses in the original text.

with averages across all language pairs used in testing.

No. of parallel versions	Average precision	
	At 0	at 1 doc.
2	0.706064	0.571491
3	0.747620	0.649269
4	0.617615	0.531873
5	0.744951	0.656140
6	0.811666	0.732602
7	0.827246	0.753070
8	0.824501	0.750000
9	0.823430	0.746053
12	0.827761	0.752632
27	0.825577	0.751316
28	0.823137	0.747807
44	0.839346	0.765789
46	0.839319	0.766667
47	0.842936	0.774561

Table 2. Summary results for all language pairs

From the above, the following should be clear: as more parallel translations are added to the index, the average precision rises considerably at first, and then begins to level off after about the seventh parallel translation. The results will of course vary according to which combination of translations is selected for the index. The number of such combinations is generally very large: for example, with 47 translations available, there are 47! / (40! 7!), or 62,891,499, possible ways of selecting 7 translations. Thus, for any particular number of parallel versions, we had to use some judgement in which parallel versions to select, since there was no way to achieve anything like exhaustive coverage of the possibilities.

Further, with more than 7 parallel translations, there is certainly no justification for saying that adding more translations or languages increases the 'noise' for languages in the test set, since beyond 7 the average precision remains fairly level. If anything, in fact, the precision still appears to rise slightly. For example, the average precision at 1 document rises by more than 0.75 percentage points between 46 and 47 versions. Given that in each of these experiments, we are measuring precision 228 times per language pair, and therefore 2,280 times in total, this small rise in precision is significant ($p \approx 0.034$). Interestingly, the 47th version to be added was parsed New Testament Greek. It appears, therefore, that the parsing helped in particular; we also have evidence from other experiments (not presented here) that overall precision is generally improved for *all* languages when Arabic wordforms are replaced by their respective citation forms (the bare root, or stem) – also a form of morphological parsing. Ancient Greek, like Arabic, is morphologically highly complex, so it would be understandable that parsing (or stemming) would help when parsing of either language is used in training.

One other point needs to be made here: the three versions added after the 44th version were the three incomplete versions (the two Greek versions cover just the New Testament, while Ancient Hebrew covers just the Old Testament). The above-mentioned increase in precision which resulted from the addition of these three versions is clear evidence that even in the case where a parallel corpus is defective for some language(s), including those languages can still result in the twofold benefit that (1) those languages are now available for analysis, and (2) precision is maintained or increased for the remaining languages.

Finally, precision at 1 document, the stricter of the two measures, is by definition less than or equal to precision at 0. This taken into account, it is also interesting that the gap between the two measures seems to narrow as more parallel translations and parsing are added, as Figure 1 shows.

For certain applications where it is important that the translation is ranked first, not just highly, among all retrieved documents, there is thus a particular benefit in using a 'massively parallel' aligned corpus.

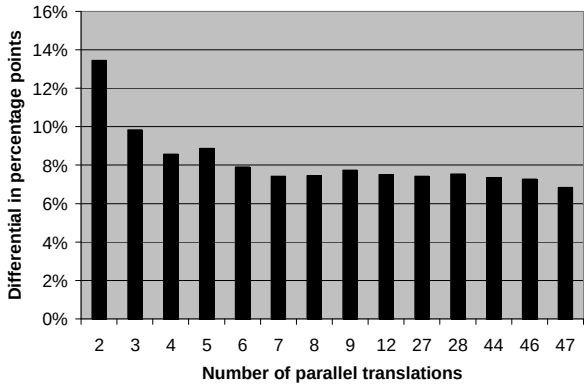


Figure 1. Differential between precision at 0 and precision at 1 document, by number of languages

Now we move on to look at more detailed results by language pair. Figure 2 below breaks down the results for precision at 1 document by language pair. In all tests, the two languages in each pair were (naturally) always included in the languages used for training. There is more volatility in the results by language pair than there is in the overall results, shown again at the right of the graph, which should come as no surprise since the averages are based on samples a tenth of the size. Generally, however, the pattern is the same for particular language pairs as it is overall; the more parallel versions are used in training, the better the average precision.

There are some more detailed observations which should also be made from Figure 2. First, the average precision clearly varies quite widely between language pairs. The language pairs with the best average precision are those in which two of English, French and Spanish are present. Of the five languages used for testing, these three cluster together genetically, since all three are Western (Germanic or Romance) Indo-European languages. Moreover, these are the three languages of the five which are written in the Roman alphabet.

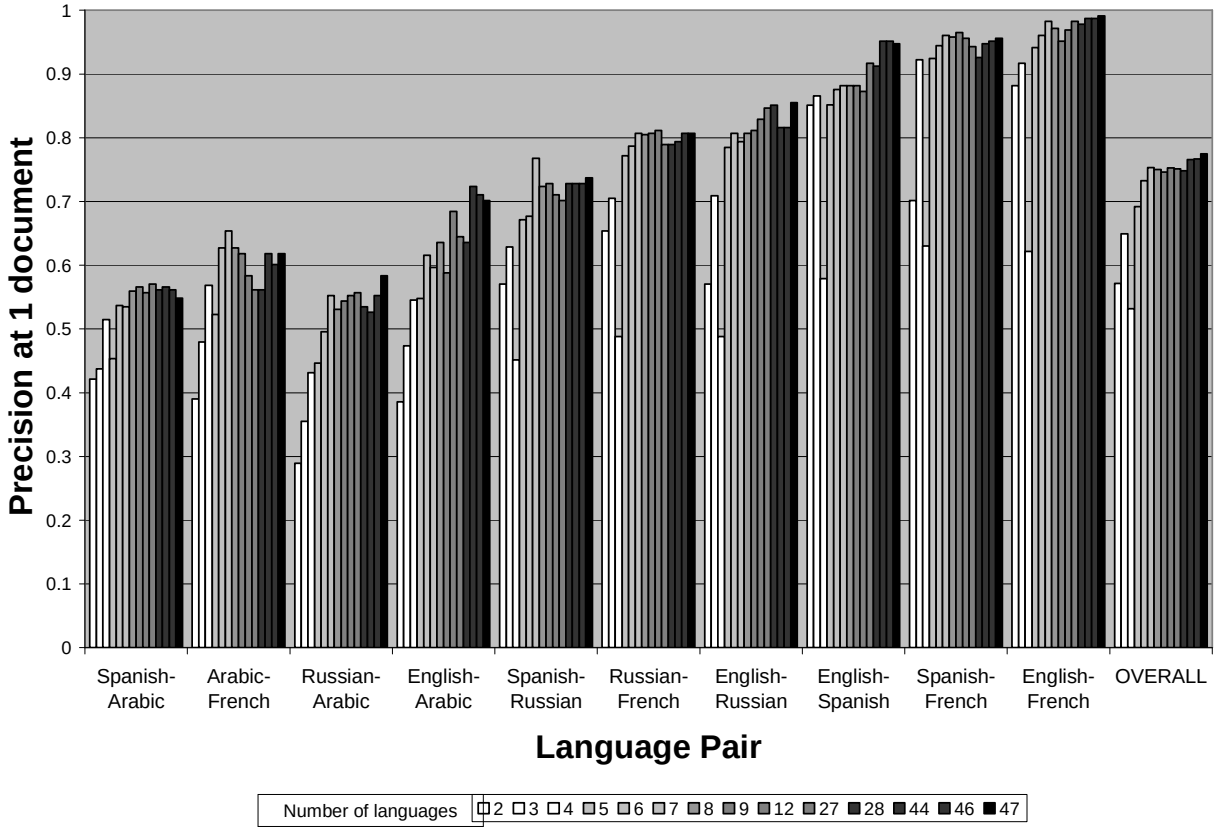


Figure 2. Chart of precision at 1 doc. by language pair and number of parallel training versions

However, we believe the explanation for the poorer results for language pairs involving either Arabic, Russian, or both, can be pinned down to something more specific. We have already partially alluded to the obvious difference between Arabic and Russian on the one hand, and English, French and Spanish on the other: that Arabic and Russian are highly morphologically rich, while English, French and Spanish are generally analytic languages. This has a clear effect on the statistics for the languages in question, as can be seen in Table 3, which is based on selected translations of the Bible for each of the languages in question.

Translation	Types	Tokens
English (King James)	12,335	789,744
Spanish (Reina Valera 1909)	28,456	704,004
Russian (Synodal 1876)	47,226	560,524
Arabic (Smith Van Dyke)	55,300	440,435
French (Darby)	20,428	812,947

Table 3. Statistics for Bible translations in 5 languages used in test data

Assuming that the respective translations are faithful (and we have no reason to believe otherwise), and based on the statistics in Table 3, it should be the case that Arabic contains the most 'information' per term (in the information theoretic sense), followed by Russian, Spanish, English and French.[3] Again, this corresponds to our intuition that much information is contained in Arabic patterns and Russian inflectional morphemes, which in English, French and Spanish would be contained in separate terms (for example, prepositions).

Without additional pre-processing, however, LSI cannot deal adequately with root-pattern or inflectional morphology. Moreover, it is clearly a weakness of LSI, or at least the standard log-entropy weighting scheme as applied within this framework, that it makes no adjustment for differences in information content per word between languages. Even though we can assume near-equivalency of information content between the different translations above, according to the standard log-entropy weighting scheme there are large differences between the total entropy of particular parallel documents; in general, languages such as English are overweighted while those such as Arabic are underweighted.

Now that this issue is in perspective, we should draw attention to another detail in Figure 2. Note that the language pairs which benefited most from the addition of Ancient Greek and Hebrew into the training data were those which included Russian, and Russian-Arabic saw the greatest increase in precision. Recall also that the 47th version to be added was the parsed Greek, so that essentially each Greek morpheme is represented by a distinct term. From Figure 2, it seems clear that the inclusion of parsed Greek in particular boosted the precision for Russian (this is most visible at the right-hand side of the set of columns for Russian-Arabic and English-Russian). There are, after all, notable similarities between modern Russian and Ancient Greek morphology (for example, the nominal case system). Essentially, the parsed Greek acts as a 'clue' to LSI in associating inflected forms in Russian with preposition/non-inflected combinations in other languages. These results seem to be further confirmation of the notion that parsing just one of the languages in the mix helps overall; the greatest boost is for those languages with morphology related to that of the parsed language, but there is at least a maintenance, and perhaps a small boost, in the precision for unrelated languages too.

Finally, we turn to look at some effects of the particular languages selected for training. Included in the results above, there were three separate tests run in which there were 6 training versions. In all three, Arabic, English, French, Russian and Spanish were included. The only factor we varied in the three tests was the sixth version. In the three tests, we used Modern Hebrew (a Semitic language, along with Arabic), Hungarian (a Uralic language, not closely related to any of the other five languages), and a second English version respectively. The results of these tests are shown in Figure 3, with figures for the test in which only 5 versions were included for comparative purposes.

From these results, it is apparent first of all that it was generally beneficial to add a sixth version, regardless of whether the version added was English, Hebrew or Hungarian. This is consistent with the results reported elsewhere in this paper. Second, it is also apparent that the greatest benefit overall was had by using an additional English version, rather than using Hebrew or Hungarian. Moreover, perhaps surprisingly, the use of Hebrew in training – even though Hebrew is related to Arabic – was of less benefit to Arabic than either Hungarian or an additional English version. It appears that the use of multiple versions in the same language is beneficial because it enables LSI to make use of the many different instantiations in the expression of a concept in a single language, and that this effect can be greater than the effect which obtains from using heterogeneous languages, even if there is a genetic relationship to existing languages.

[3] To clarify the meaning of 'term' here: for all languages except Chinese, text is tokenized in our framework into terms using regular expressions; each non-word character (such as punctuation or white space) is assumed to mark the boundary of a word. For Chinese, we made the simplifying assumption that each character represented a separate term.

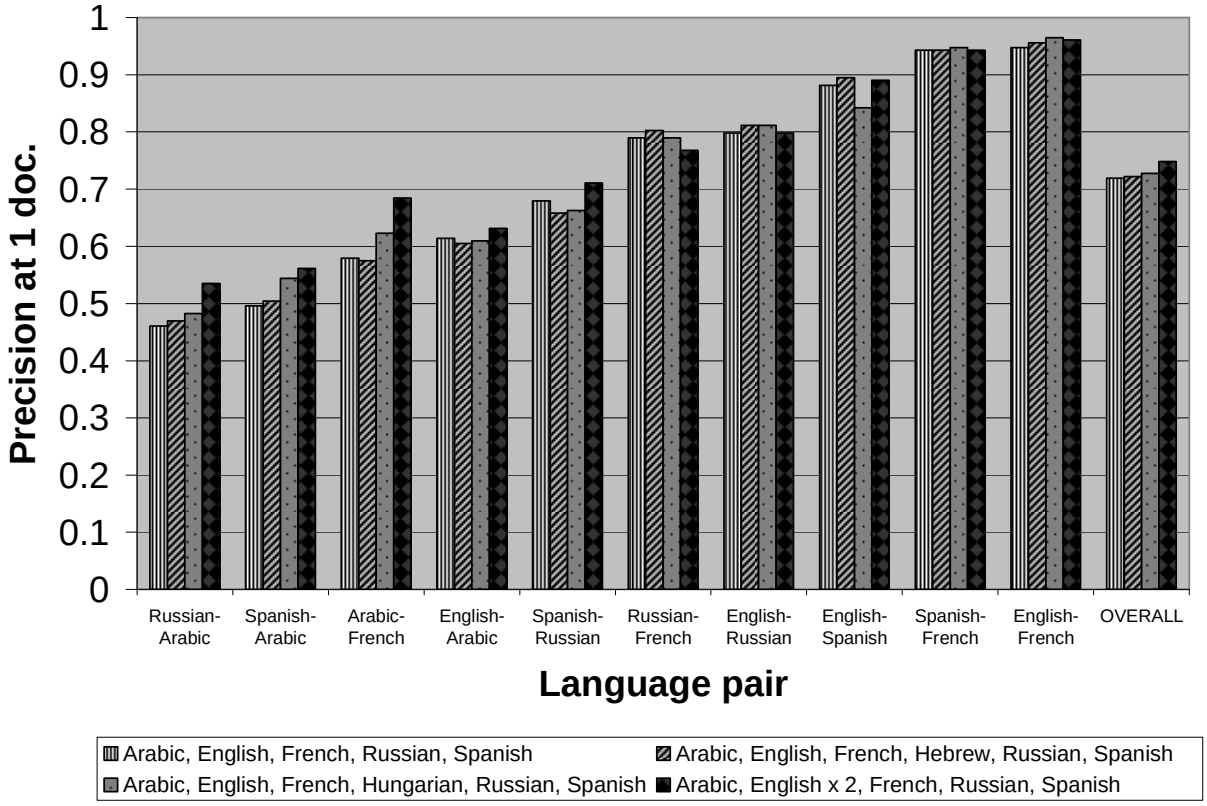


Figure 3. Precision at 1 document for 6 training versions, with results of using different mixes of languages for training

Figure 3 may also shed some additional light on one other detail from Figure 2: a perceptible jump in precision between 28 and 44 training versions for Arabic-English and Arabic-French. It should be mentioned that among the 16 additional versions were five English versions (American Standard Version, Basic English Bible, Darby, Webster's Bible, and Young's Literal Translation), and one French version (Louis Segond 1910). It seems that Figure 2 and Figure 3 both point to the same thing: that the use of parallel versions or translations in a *single* language can be particularly beneficial to overall precision within the LSI framework – even to a greater extent than the use of parallel translations in different languages.

5 Conclusion

In this paper, we have shown how 'massive parallelism' in an aligned corpus can be used to improve the results of cross-language information retrieval. Apart from the obvious advantage (the ability to automate the processing of a greater variety of linguistic data within a single framework), we have shown that including more parallel translations in training improves the precision of CLIR across the board. This is true whether the additional translations are in the language of another translation already within the training set, whether they are in a related language, or whether they are in an unrelated language; although this is not to say that these choices do not lead to (generally minor) variations in the results. The improvement in precision also appears to hold whether the additional translations are complete or incomplete, and it appears that morphological pre-processing helps, not just for the languages pre-processed, but again across the board.

Our work also offers further evidence that the supply of useful pre-existing parallel corpora is not perhaps as scarce as it is sometimes claimed to be. Compilation of the 47-version parallel corpus we used was not very time-consuming, especially if the time taken to clean the data is not taken into

account, and all the textual material we used is publicly available on the World Wide Web.

While the experiments we performed were on non-standard test collections (primarily because the Qu'ran was easy to obtain in multiple languages), it seems that there is no reason to believe our general observation – that more parallelism in the training data is beneficial for cross-language retrieval – would not hold for text from other domains. Whether the genre of text used as training data affects the *absolute* rate of retrieval precision for text of a different genre (e.g. news articles, shopping websites) is a separate question, and one we intend to address more fully in future work.

In summary, it appears that we are able to achieve the results we do partly because of the inherent properties of LSI. In essence, when the data from more and more parallel translations are subjected to SVD, the LSI 'concepts' become more and more reinforced. The resulting trend for precision to increase, despite 'blips' for individual languages, can be seen for all languages. To put it in more prosaic terms, the more different ways the same things are said in, the more understandable they become – including in cross-language information retrieval.

Acknowledgement

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

Lars Asker. 2004. Building Resources: Experiences from Amharic Cross Language Information Retrieval. Paper presented at *Cross-Language Information Retrieval and Evaluation: Workshop of the Cross-Language Evaluation Forum, CLEF 2004*.

Ricardo Baeza-Yates and Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval*. New York: ACM Press.

Michael Berry, Theresa Do, Gavin O'Brien, Vijay Krishna, and Sowmimi Varadhan. 1996. SVDPACKC (Version 1.0) User's Guide. Knoxville, TN: University of Tennessee.

Bible Society. 2006. *A Statistical Summary of Languages with the Scriptures*. Accessed at http://www.biblesociety.org/latestnews/latest341-slr2005stats.html on Jan. 5, 2007.

Biola University. 2005-2006. *The Unbound Bible*. Accessed at http://www.unboundbible.com/ on Jan. 5, 2007.

Peter Chew, Stephen Verzi, Travis Bauer and Jonathan McClain. 2006. Evaluation of the Bible as a Resource for Cross-Language Information Retrieval. In *Proceedings of the Workshop on Multilingual Language Resources and Interoperability*, 68-74. Sydney: Association for Computational Linguistics.

Susan Dumais. 1991. Improving the Retrieval of Information from External Sources. *Behavior Research Methods, Instruments, and Computers* 23(2):229-236.

Julio Gonzalo. 2001. Language Resources in Cross-Language Text Retrieval: a CLEF Perspective. In Carol Peters (ed.). *Cross-Language Information Retrieval and Evaluation: Workshop of the Cross-Language Evaluation Forum, CLEF 2000*: 36-47. Berlin: Springer-Verlag.

Dragos Munteanu and Daniel Marcu. 2006. Improving Machine Translation Performance by Exploiting Non-Parallel Corpora. *Computational Linguistics* 31(4):477-504.

Jian-Yun Nie and Fuman Jin. 2002. A Multilingual Approach to Multilingual Information Retrieval. *Proceedings of the Cross-Language Evaluation Forum*, 101-110. Berlin: Springer-Verlag.

Jian-Yun Nie, Michel Simard, Pierre Isabelle, and Richard Durand. 1999. Cross-Language Retrieval based on Parallel Texts and Automatic Mining of Parallel Texts from the Web. *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 74-81, August 15-19, 1999, Berkeley, CA.

Carol Peters (ed.). 2001. *Cross-Language Information Retrieval and Evaluation: Workshop of the Cross-Language Evaluation Forum, CLEF 2000*. Berlin: Springer-Verlag.

Recherche appliquée en linguistique informatique (RALI). 2006. *Corpus aligné bilingue anglais-français*. Accessed at http://rali.iro.umontreal.ca/ on February 22, 2006.

Philip Resnik, Mari Broman Olsen, and Mona Diab. 1999. The Bible as a Parallel Corpus: Annotating the "Book of 2000 Tongues". *Computers and the Humanities*, 33: 129-153.

A Re-examination of Machine Learning Approaches for Sentence-Level MT Evaluation

Joshua S. Albrecht and Rebecca Hwa
Department of Computer Science
University of Pittsburgh
{jsa8,hwa}@cs.pitt.edu

Abstract

Recent studies suggest that machine learning can be applied to develop good automatic evaluation metrics for machine translated sentences. This paper further analyzes aspects of learning that impact performance. We argue that previously proposed approaches of training a *Human-Likeness classifier* is not as well correlated with human judgments of translation quality, but that *regression-based learning* produces more reliable metrics. We demonstrate the feasibility of regression-based metrics through empirical analysis of learning curves and generalization studies and show that they can achieve higher correlations with human judgments than standard automatic metrics.

1 Introduction

As machine translation (MT) research advances, the importance of its evaluation also grows. Efficient evaluation methodologies are needed both for facilitating the system development cycle and for providing an unbiased comparison between systems. To this end, a number of automatic evaluation metrics have been proposed to approximate human judgments of MT output quality. Although studies have shown them to correlate with human judgments at the document level, they are not sensitive enough to provide reliable evaluations at the sentence level (Blatz et al., 2003). This suggests that current metrics do not fully reflect the set of criteria that people use in judging sentential translation quality.

A recent direction in the development of metrics for sentence-level evaluation is to apply machine learning to create an improved composite metric out of less indicative ones (Corston-Oliver et al., 2001; Kulesza and Shieber, 2004). Under the assumption that good machine translation will produce "human-like" sentences, classifiers are trained to predict whether a sentence is authored by a human or by a machine based on features of that sentence, which may be the sentence's scores from individual automatic evaluation metrics. The confidence of the classifier's prediction can then be interpreted as a judgment on the translation quality of the sentence. Thus, the composite metric is encoded in the confidence scores of the classification labels.

While the learning approach to metric design offers the promise of ease of combining multiple metrics and the potential for improved performance, several salient questions should be addressed more fully. First, is learning a "Human Likeness" classifier the most suitable approach for framing the MT-evaluation question? An alternative is regression, in which the composite metric is explicitly learned as a function that approximates humans' quantitative judgments, based on a set of human evaluated training sentences. Although regression has been considered on a small scale for a single system as confidence estimation (Quirk, 2004), this approach has not been studied as extensively due to scalability and generalization concerns. Second, how does the diversity of the model features impact the learned metric? Third, how well do learning-based metrics generalize beyond their training examples? In particular, how well can a metric that was developed based

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 880–887,
Prague, Czech Republic, June 2007.

on one group of MT systems evaluate the translation qualities of new systems?

In this paper, we argue for the viability of a regression-based framework for sentence-level MT-evaluation. Through empirical studies, we first show that having an accurate Human-Likeness classifier does not necessarily imply having a good MT-evaluation metric. Second, we analyze the resource requirement for regression models for different sizes of feature sets through learning curves. Finally, we show that SVM-regression metrics generalize better than SVM-classification metrics in their evaluation of systems that are different from those in the training set (by languages and by years), and their correlations with human assessment are higher than standard automatic evaluation metrics.

2 MT Evaluation

Recent automatic evaluation metrics typically frame the evaluation problem as a comparison task: how *similar* is the machine-produced output to a set of human-produced reference translations for the same source text? However, as the notion of similarity is itself underspecified, several different families of metrics have been developed. First, similarity can be expressed in terms of string edit distances. In addition to the well-known word error rate (WER), more sophisticated modifications have been proposed (Tillmann et al., 1997; Snover et al., 2006; Leusch et al., 2006). Second, similarity can be expressed in terms of common word sequences. Since the introduction of BLEU (Papineni et al., 2002) the basic n-gram precision idea has been augmented in a number of ways. Metrics in the Rouge family allow for skip n-grams (Lin and Och, 2004a); Kauchak and Barzilay (2006) take paraphrasing into account; metrics such as METEOR (Banerjee and Lavie, 2005) and GTM (Melamed et al., 2003) calculate both recall and precision; METEOR is also similar to SIA (Liu and Gildea, 2006) in that word class information is used. Finally, researchers have begun to look for similarities at a deeper structural level. For example, Liu and Gildea (2005) developed the Sub-Tree Metric (STM) over constituent parse trees and the Head-Word Chain Metric (HWCM) over dependency parse trees.

With this wide array of metrics to choose from, MT developers need a way to evaluate them. One possibility is to examine whether the automatic metric ranks the human reference translations highly with respect to machine translations (Lin and Och, 2004b; Amigó et al., 2006). The reliability of a metric can also be more directly assessed by determining how well it correlates with human judgments of the same data. For instance, as a part of the recent NIST sponsored MT Evaluation, each translated sentence by participating systems is evaluated by two (non-reference) human judges on a five point scale for its *adequacy* (does the translation retain the meaning of the original source text?) and *fluency* (does the translation sound natural in the target language?). These human assessment data are an invaluable resource for measuring the reliability of automatic evaluation metrics. In this paper, we show that they are also informative in developing better metrics.

3 MT Evaluation with Machine Learning

A good automatic evaluation metric can be seen as a computational model that captures a human's decision process in making judgments about the adequacy and fluency of translation outputs. Inferring a cognitive model of human judgments is a challenging problem because the ultimate judgment encompasses a multitude of fine-grained decisions, and the decision process may differ slightly from person to person. The metrics cited in the previous section aim to capture certain aspects of human judgments. One way to combine these metrics in a uniform and principled manner is through a learning framework. The individual metrics participate as input features, from which the learning algorithm infers a composite metric that is optimized on training examples.

Reframing sentence-level translation evaluation as a classification task was first proposed by Corston-Oliver et al. (2001). Interestingly, instead of recasting the classification problem as a "Human Acceptability" test (distinguishing good translations outputs from bad one), they chose to develop a Human-Likeness classifier (distinguishing outputs seem human-produced from machine-produced ones) to avoid the necessity of obtaining manually labeled training examples. Later, Kulesza and Shieber (2004) noted that if a classifier provides a

confidence score for its output, that value can be interpreted as a quantitative estimate of the input instance's translation quality. In particular, they trained an SVM classifier that makes its decisions based on a set of input features computed from the sentence to be evaluated; the distance between input feature vector and the separating hyperplane then serves as the evaluation score. The underlying assumption for both is that improving the accuracy of the classifier on the Human-Likeness test will also improve the implicit MT evaluation metric.

A more direct alternative to the classification approach is to learn via regression and explicitly optimize for a function (i.e. MT evaluation metric) that approximates human judgments in training examples. Kulesza and Shieber (2004) raised two main objections against regression for MT evaluations. One is that regression requires a large set of labeled training examples. Another is that regression may not generalize well over time, and re-training may become necessary, which would require collecting additional human assessment data. While these are legitimate concerns, we show through empirical studies (in Section 4.2) that the additional resource requirement is not impractically high, and that a regression-based metric has higher correlations with human judgments and generalizes better than a metric derived from a Human-Likeness classifier.

3.1 Relationship between Classification and Regression

Classification and regression are both processes of function approximation; they use training examples as sample instances to learn the mapping from inputs to the desired outputs. The major difference between classification and regression is that the function learned by a classifier is a set of decision boundaries by which to classify its inputs; thus its outputs are discrete. In contrast, a regression model learns a continuous function that directly maps an input to a continuous value. An MT evaluation metric is inherently a continuous function. Casting the task as a 2-way classification may be too coarse-grained. The Human-Likeness formulation of the problem introduces another layer of approximation by assuming equivalence between "Like Human-Produced" and "Well-formed" sentences. In Section 4.1, we show empirically that high accuracy in the Human-Likeness test does not necessarily entail good MT evaluation judgments.

3.2 Feature Representation

To ascertain the resource requirements for different model sizes, we considered two feature models. The smaller one uses the same nine features as Kulesza and Shieber, which were derived from BLEU and WER. The full model consists of 53 features: some are adapted from recently developed metrics; others are new features of our own. They fall into the following major categories[1]:

String-based metrics over references These include the nine Kulesza and Shieber features as well as precision, recall, and fragmentation, as calculated in METEOR; ROUGE-inspired features that are non-consecutive bigrams with a gap size of m, where $1 \leq m \leq 5$ (skip-m-bigram), and ROUGE-L (longest common subsequence).

Syntax-based metrics over references We unrolled HWCM into their individual chains of length c (where $2 \leq c \leq 4$); we modified STM so that it is computed over unlexicalized constituent parse trees as well as over dependency parse trees.

String-based metrics over corpus Features in this category are similar to those in *String-based metric over reference* except that a large English corpus is used as "reference" instead.

Syntax-based metrics over corpus A large dependency treebank is used as the "reference" instead of parsed human translations. In addition to adaptations of the *Syntax-based metrics over references*, we have also created features to verify the argument structures for certain syntactic categories.

4 Empirical Studies

In these studies, the learning models used for both classification and regression are support vector machines (SVM) with Gaussian kernels. All models are trained with SVM-Light (Joachims, 1999). Our primary experimental dataset is from NIST's 2003

[1] As feature engineering is not the primary focus of this paper, the features are briefly described here, but implementational details will be made available in a technical report.

Chinese MT Evaluations, in which the fluency and adequacy of 919 sentences produced by six MT systems are scored by two human judges on a 5-point scale[2]. Because the judges evaluate sentences according to their individual standards, the resulting scores may exhibit a biased distribution. We normalize human judges' scores following the process described by Blatz et al. (2003). The overall human assessment score for a translation output is the average of the sum of two judges' normalized fluency and adequacy scores. The full dataset ($6 \times 919 = 5514$ instances) is split into sets of training, heldout and test data. Heldout data is used for parameter tuning (i.e., the slack variable and the width of the Gaussian). When training classifiers, assessment scores are not used, and the training set is augmented with all available human reference translation sentences ($4 \times 919 = 3676$ instances) to serve as positive examples.

To judge the quality of a metric, we compute Spearman rank-correlation coefficient, which is a real number ranging from -1 (indicating perfect negative correlations) to +1 (indicating perfect positive correlations), between the metric's scores and the averaged human assessments on test sentences. We use Spearman instead of Pearson because it is a distribution-free test. To evaluate the relative reliability of different metrics, we use bootstrapping re-sampling and paired t-test to determine whether the difference between the metrics' correlation scores has statistical significance (at 99.8% confidence level)(Koehn, 2004). Each reported correlation rate is the average of 1000 trials; each trial consists of n sampled points, where n is the size of the test set. Unless explicitly noted, the qualitative differences between metrics we report are statistically significant. As a baseline comparison, we report the correlation rates of three standard automatic metrics: BLEU, METEOR, which incorporates recall and stemming, and HWCM, which uses syntax. BLEU is smoothed to be more appropriate for sentence-level evaluation (Lin and Och, 2004b), and the bigram versions of BLEU and HWCM are reported because they have higher correlations than when longer n-grams are included. This phenomenon has been previously observed by Liu and Gildea (2005).

[2]This corpus is available from the Linguistic Data Consortium as Multiple Translation Chinese Part 4.

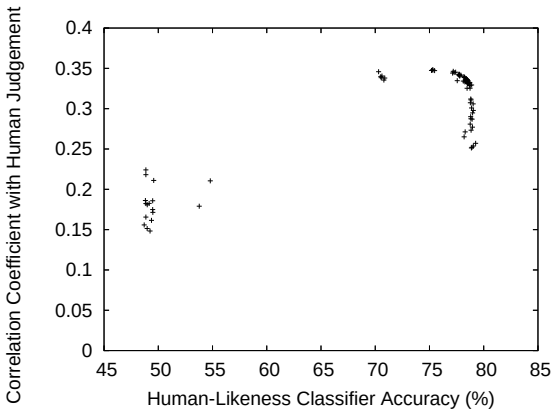


Figure 1: This scatter plot compares classifiers' accuracy with their corresponding metrics' correlations with human assessments

4.1 Relationship between Classification Accuracy and Quality of Evaluation Metric

A concern in using a metric derived from a Human-Likeness classifier is whether it would be predictive for MT evaluation. Kulesza and Shieber (2004) tried to demonstrate a positive correlation between the Human-Likeness classification task and the MT evaluation task empirically. They plotted the classification accuracy and evaluation reliability for a number of classifiers, which were generated as a part of a greedy search for kernel parameters and found some linear correlation between the two. This proof of concept is a little misleading, however, because the population of the sampled classifiers was biased toward those from the same neighborhood as the local optimal classifier (so accuracy and correlation may only exhibit linear relationship locally). Here, we perform a similar study except that we sampled the kernel parameter more uniformly (on a log scale). As Figure 1 confirms, having an accurate Human-Likeness classifier does not necessarily entail having a good MT evaluation metric. Although the two tasks do seem to be positively related, and in the limit there may be a system that is good at both tasks, one may improve classification without improving MT evaluation. For this set of heldout data, at the near 80% accuracy range, a derived metric might have an MT evaluation correlation coefficient anywhere between 0.25 (on par with

unsmoothed BLEU, which is known to be unsuitable for sentence-level evaluation) and 0.35 (competitive with standard metrics).

4.2 Learning Curves

To investigate the feasibility of training regression models from assessment data that are currently available, we consider both a small and a large regression model. The smaller model consists of nine features (same as the set used by Kulesza and Shieber); the other uses the full set of 53 features as described in Section 3.2. The reliability of the trained metrics are compared with those developed from Human-Likeness classifiers. We follow a similar training and testing methodology as previous studies: we held out 1/6 of the assessment dataset for SVM parameter tuning; five-fold cross validation is performed with the remaining sentences. Although the metrics are evaluated on unseen test sentences, the sentences are produced by the same MT systems that produced the training sentences. In later experiments, we investigate generalizing to more distant MT systems.

Figure 2(a) shows the learning curves for the two regression models. As the graph indicates, even with a limited amount of human assessment data, regression models can be trained to be comparable to standard metrics (represented by METEOR in the graph). The small feature model is close to convergence after 1000 training examples[3]. The model with a more complex feature set does require more training data, but its correlation began to overtake METEOR after 2000 training examples. This study suggests that the start-up cost of building even a moderately complex regression model is not impossibly high.

Although we cannot directly compare the learning curves of the Human-Likeness classifiers to those of the regression models (since the classifier's training examples are automatically labeled), training examples for classifiers are not entirely free: human reference translations still must be developed for the source sentences. Figure 2(c) shows the learning curves for training Human-Likeness classifiers (in terms of improving a classifier's accuracy) using the same two feature sets, and Figure 2(b) shows the correlations of the metrics derived from the corresponding classifiers. The pair of graphs show, especially in the case of the larger feature set, that a large improvement in classification accuracy does not bring proportional improvement in its corresponding metrics's correlation; with an accuracy of near 90%, its correlation coefficient is 0.362, well below METEOR.

This experiment further confirms that judging Human-Likeness and judging Human-Acceptability are not tightly coupled. Earlier, we have shown in Figure 1 that different SVM parameterizations may result in classifiers with the same accuracy rate but different correlations rates. As a way to incorporate some assessment information into classification training, we modify the parameter tuning process so that SVM parameters are chosen to optimize for assessment correlations in the heldout data. By incurring this small amount of human assessed data, this parameter search improves the classifier's correlations: the metric using the smaller feature set increased from 0.423 to 0.431, and that of the larger set increased from 0.361 to 0.422.

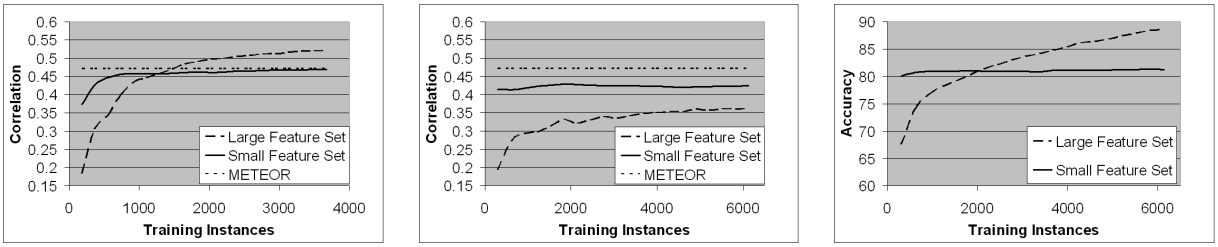
4.3 Generalization

We conducted two generalization studies. The first investigates how well the trained metrics evaluate systems from other years and systems developed for a different source language. The second study delves more deeply into how variations in the training examples affect a learned metric's ability to generalize to distant systems. The learning models for both experiments use the full feature set.

Cross-Year Generalization To test how well the learning-based metrics generalize to systems from different years, we trained both a regression-based metric (R03) and a classifier-based metric (C03) with the entire NIST 2003 Chinese dataset (using 20% of the data as heldout[4]). All metrics are then applied to three new datasets: NIST 2002 Chinese MT Evaluation (3 systems, 2634 sentences total), NIST 2003 Arabic MT Evaluation (2 systems, 1326 sentences total), and NIST 2004 Chinese MT Evaluation (10 systems, 4470 sentences total). The results

[3]The total number of labeled examples required is closer to 2000, since the heldout set uses 919 labeled examples.

[4]Here, too, we allowed the classifier's parameters to be tuned for correlation with human assessment on the heldout data rather than accuracy.



(a) (b) (c)

Figure 2: Learning curves: (a) correlations with human assessment using regression models; (b) correlations with human assessment using classifiers; (c) classifier accuracy on determining Human-Likeness.

Dataset	R03	C03	BLEU	MET.	HWCM
2002 Ara	**0.466**	0.384	0.423	0.431	0.424
2002 Chn	**0.309**	0.250	0.269	0.290	0.260
2004 Chn	**0.602**	0.566	0.588	0.563	0.546

Table 1: Correlations for cross-year generalization. Learning-based metrics are developed from NIST 2003 Chinese data. All metrics are tested on datasets from 2003 Arabic, 2002 Chinese and 2004 Chinese.

are summarized in Table 1. We see that R03 consistently has a better correlation rate than the other metrics.

At first, it may seem as if the difference between R03 and BLEU is not as pronounced for the 2004 dataset, calling to question whether a learned metric might become quickly out-dated, we argue that this is not the case. The 2004 dataset has many more participating systems, and they span a wider range of qualities. Thus, it is easier to achieve a high rank correlation on this dataset than previous years because most metrics can qualitatively discern that sentences from one MT system are better than those from another. In the next experiment, we examine the performance of R03 with respect to each MT system in the 2004 dataset and show that its correlation rate is higher for better MT systems.

Relationship between Training Examples and Generalization Table 2 shows the result of a generalization study similar to before, except that correlations are performed on each system. The rows order the test systems by their translation qualities from the best performing system (2004-Chn1, whose average human assessment score is 0.655 out of 1.0) to the worst (2004-Chn10, whose score is 0.255). In addition to the regression metric from the previous experiment (R03-all), we consider two more regression metrics trained from subsets of the 2003 dataset: R03-Bottom5 is trained from the subset that excludes the best 2003 MT system, and R03-Top5 is trained from the subset that excludes the worst 2003 MT system.

We first observe that on a per test-system basis, the regression-based metrics generally have better correlation rates than BLEU, and that the gap is as wide as what we have observed in the earlier cross-years studies. The one exception is when evaluating 2004-Chn8. None of the metrics seems to correlate very well with human judges on this system. Because the regression-based metric uses these individual metrics as features, its correlation also suffers.

During regression training, the metric is optimized to minimize the difference between its prediction and the human assessments of the training data. If the input feature vector of a test instance is in a very distant space from training examples, the chance for error is higher. As seen from the results, the learned metrics typically perform better when the training examples include sentences from higher-quality systems. Consider, for example, the differences between R03-all and R03-Top5 versus the differences between R03-all and R03-Bottom5. Both R03-Top5 and R03-Bottom5 differ from R03-all by one subset of training examples. Since R03-all's correlation rates are generally closer to R03-Top5 than to R03-Bottom5, we see that having seen extra training examples from a bad system is not as harmful as having not seen training examples from a good system. This is expected, since there are many ways to create bad translations, so seeing a partic-

	R03-all	R03-Bottom5	R03-Top5	BLEU	METEOR	HWCM
2004-Chn1	0.495	0.460	**0.518**	0.456	0.457	0.444
2004-Chn2	0.398	0.330	**0.440**	0.352	0.347	0.344
2004-Chn3	0.425	0.389	**0.459**	0.369	0.402	0.369
2004-Chn4	**0.432**	0.392	0.434	0.400	0.400	0.362
2004-Chn5	**0.452**	0.441	0.443	0.370	0.426	0.326
2004-Chn6	*0.405*	0.392	**0.406**	0.390	0.357	0.380
2004-Chn7	0.443	0.432	**0.448**	0.390	0.408	0.392
2004-Chn8	0.237	0.256	0.256	**0.265**	0.259	0.179
2004-Chn9	0.581	0.569	**0.591**	0.527	0.537	0.535
2004-Chn10	0.314	0.313	**0.354**	0.321	0.303	0.358
2004-all	0.602	0.567	**0.617**	0.588	0.563	0.546

Table 2: Metric correlations within each system. The columns specify which metric is used. The rows specify which MT system is under evaluation; they are ordered by human-judged system quality, from best to worst. For each evaluated MT system (row), the highest coefficient in bold font, and those that are statistically comparable to the highest are shown in italics.

ular type of bad translations from one system may not be very informative. In contrast, the neighborhood of good translations is much smaller, and is where all the systems are aiming for; thus, assessments of sentences from a good system can be much more informative.

4.4 Discussion

Experimental results confirm that learning from training examples that have been doubly approximated (class labels instead of ordinals, human-likeness instead of human-acceptability) does negatively impact the performance of the derived metrics. In particular, we showed that they do not generalize as well to new data as metrics trained from direct regression.

We see two lingering potential objections toward developing metrics with regression-learning. One is the concern that a system under evaluation might try to explicitly "game the metric[5]." This is a concern shared by all automatic evaluation metrics, and potential problems in stand-alone metrics have been analyzed (Callison-Burch et al., 2006). In a learning framework, potential pitfalls for individual metrics are ameliorated through a combination of evidences. That said, it is still prudent to defend against the potential of a system gaming a subset of the features. For example, our fluency-predictor features are not strong indicators of translation qualities by themselves. We want to avoid training a metric that assigns a higher than deserving score to a sentence that just happens to have many n-gram matches against the target-language reference corpus. This can be achieved by supplementing the current set of human assessed training examples with automatically assessed training examples, similar to the labeling process used in the Human-Likeness classification framework. For instance, as negative training examples, we can incorporate fluent sentences that are not adequate translations and assign them low overall assessment scores.

A second, related concern is that because the metric is trained on examples from current systems using currently relevant features, even though it generalizes well in the near term, it may not continue to be a good predictor in the distant future. While periodic retraining may be necessary, we see value in the flexibility of the learning framework, which allows for new features to be added. Moreover, adaptive learning methods may be applicable if a small sample of outputs of some representative translation systems is manually assessed periodically.

5 Conclusion

Human judgment of sentence-level translation quality depends on many criteria. Machine learning affords a unified framework to compose these criteria into a single metric. In this paper, we have demonstrated the viability of a regression approach to learning the composite metric. Our experimental results show that by training from some human as-

[5]Or, in a less adversarial setting, a system may be performing minimum error-rate training (Och, 2003)

sessments, regression methods result in metrics that have better correlations with human judgments even as the distribution of the tested population changes.

Acknowledgments

This work has been supported by NSF Grants IIS-0612791 and IIS-0710695. We would like to thank Regina Barzilay, Ric Crabbe, Dan Gildea, Alex Kulesza, Alon Lavie, and Matthew Stone as well as the anonymous reviewers for helpful comments and suggestions. We are also grateful to NIST for making their assessment data available to us.

References

Enrique Amigó, Jesús Giménez, Julio Gonzalo, and Lluís Màrquez. 2006. MT evaluation: Human-like vs. human acceptable. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, Sydney, Australia, July.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for MT evaluation with improved correlation with human judgments. In *ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, June.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2003. Confidence estimation for machine translation. Technical Report Natural Language Engineering Workshop Final Report, Johns Hopkins University.

Christopher Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of BLEU in machine translation research. In *The Proceedings of the Thirteenth Conference of the European Chapter of the Association for Computational Linguistics*.

Simon Corston-Oliver, Michael Gamon, and Chris Brockett. 2001. A machine learning approach to the automatic evaluation of machine translation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, July.

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.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for automatic evaluation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, New York City, USA, June.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP-04)*.

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)*, Baltimore, MD, October.

Gregor Leusch, Nicola Ueffing, and Hermann Ney. 2006. CDER: Efficient MT evaluation using block movements. In *The Proceedings of the Thirteenth Conference of the European Chapter of the Association for Computational Linguistics*.

Chin-Yew Lin and Franz Josef Och. 2004a. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, July.

Chin-Yew Lin and Franz Josef Och. 2004b. Orange: a method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, August.

Ding Liu and Daniel Gildea. 2005. Syntactic features for evaluation of machine translation. In *ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, June.

Ding Liu and Daniel Gildea. 2006. Stochastic iterative alignment for machine translation evaluation. In *Proceedings of the Joint Conference of the International Conference on Computational Linguistics and the Association for Computational Linguistics (COLING-ACL'2006) Poster Session*, July.

I. Dan Melamed, Ryan Green, and Joseph Turian. 2003. Precision and recall of machine translation. In *In Proceedings of the HLT-NAACL 2003: Short Papers*, pages 61–63, Edmonton, Alberta.

Franz Josef Och. 2003. Minimum error rate training for statistical machine translation. In *Proceedings of the 41st Annual Meeting of the 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 *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, PA.

Christopher Quirk. 2004. Training a sentence-level machine translation confidence measure. In *Proceedings of LREC 2004*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 8th Conference of the Association for Machine Translation in the Americas (AMTA-2006)*.

Christoph Tillmann, Stephan Vogel, Hermann Ney, Hassan Sawaf, and Alex Zubiaga. 1997. Accelerated DP-based search for statistical translation. In *Proceedings of the 5th European Conference on Speech Communication and Technology (EuroSpeech '97)*.

Automatic Acquisition of Ranked Qualia Structures from the Web[1]

Philipp Cimiano
Inst. AIFB, University of Karlsruhe
Englerstr. 11, D-76131 Karlsruhe
cimiano@aifb.uni-karlsruhe.de

Johanna Wenderoth
Inst. AIFB, University of Karlsruhe
Englerstr. 11, D-76131 Karlsruhe
jowenderoth@googlemail.com

Abstract

This paper presents an approach for the automatic acquisition of qualia structures for nouns from the Web and thus opens the possibility to explore the impact of qualia structures for natural language processing at a larger scale. The approach builds on earlier work based on the idea of matching specific lexico-syntactic patterns conveying a certain semantic relation on the World Wide Web using standard search engines. In our approach, the qualia elements are actually ranked for each qualia role with respect to some measure. The specific contribution of the paper lies in the extensive analysis and quantitative comparison of different measures for ranking the qualia elements. Further, for the first time, we present a quantitative evaluation of such an approach for learning qualia structures with respect to a handcrafted gold standard.

1 Introduction

Qualia structures have been originally introduced by (Pustejovsky, 1991) and are used for a variety of purposes in natural language processing (NLP), such as for the analysis of compounds (Johnston and Busa, 1996) as well as co-composition and coercion (Pustejovsky, 1991), but also for bridging reference resolution (Bos et al., 1995). Further, it has also been argued that qualia structures and lexical semantic relations in general have applications in information retrieval (Voorhees, 1994; Pustejovsky et al., 1993). One major bottleneck however is that currently qualia structures need to be created by hand, which is probably also the reason why there are almost no practical NLP systems using qualia structures, but a lot of systems relying on publicly available resources such as WordNet (Fellbaum, 1998) or FrameNet (Baker et al., 1998) as source of lexical/world knowledge. The work described in this paper addresses this issue and presents an approach to automatically learning qualia structures for nouns from the Web. The approach is inspired in recent work on using the Web to identify instances of a relation of interest such as in (Markert et al., 2003) and (Etzioni et al., 2005). These approaches rely on a combination of the usage of lexico-syntactic pattens conveying a certain relation of interest as described in (Hearst, 1992) with the idea of using the web as a big corpus (cf. (Kilgariff and Grefenstette, 2003)).
Our approach directly builds on our previous work (Cimiano and Wenderoth, 2005) an adheres to the principled idea of learning ranked qualia structures. In fact, a ranking of qualia elements is useful as it helps to determine a cut-off point and as a reliability indicator for lexicographers inspecting the qualia structures. In contrast to our previous work, the focus of this paper lies in analyzing different measures for ranking the qualia elements in the automatically acquired qualia structures. We also introduce additional patterns for the agentive role which make use of wildcard operators. Further, we present a gold standard for qualia structures created for the 30 words used in the evaluation of Yamada and Baldwin (Yamada and Baldwin, 2004). The evaluation

[1]The work reported in this paper has been supported by the X-Media project, funded by the European Commission under EC grant number IST-FP6-026978 as well by the SmartWeb project, funded by the German Ministry of Research. Thanks to all our colleagues for helping to evaluate the approach.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 888–895,
Prague, Czech Republic, June 2007.

presented here is thus much more extensive than our previous one (Cimiano and Wenderoth, 2005), in which only 7 words were used. We present a quantitative evaluation of our approach and a comparison of the different ranking measures with respect to this gold standard. Finally, we also provide an evaluation in which test persons were asked to inspect and rate the learned qualia structures a posteriori. The paper is structured as follows: Section 2 introduces qualia structures for the sake of completeness and describes the specific structures we aim to acquire. Section 3 describes our approach in detail, while Section 4 discusses the ranking measures used. Section 5 then presents the gold standard as well as the qualitative evaluation of our approach. Before concluding, we discuss related work in Section 6.

2 Qualia Structures

In the Generative Lexicon (GL) framework (Pustejovsky, 1991), Pustejovsky reused Aristotle's basic factors (i.e. the material, agentive, formal and final causes) for the description of the meaning of lexical elements. In fact, he introduced so called *qualia structures* by which the meaning of a lexical element is described in terms of four roles: *Constitutive* (describing physical properties of an object, i.e. its weight, material as well as parts and components), *Agentive* (describing factors involved in the bringing about of an object, i.e. its creator or the causal chain leading to its creation), *Formal* (describing properties which distinguish an object within a larger domain, i.e. orientation, magnitude, shape and dimensionality), and *Telic* (describing the purpose or function of an object).

Most of the qualia structures used in (Pustejovsky, 1991) however seem to have a more restricted interpretation. In fact, in most examples the *Constitutive* role seems to describe the parts or components of an object, while the *Agentive* role is typically described by a verb denoting an action which typically brings the object in question into existence. The *Formal* role normally consists in typing information about the object, i.e. its hypernym. In our approach, we aim to acquire qualia structures according to this restricted interpretation.

3 Automatically Acquiring Qualia Structures

Our approach to learning qualia structures from the Web is on the one hand based on the assumption that instances of a certain semantic relation can be acquired by matching certain lexico-syntactic patterns more or less reliably conveying the relation of interest in line with the seminal work of Hearst (Hearst, 1992), who defined patterns conveying hyponym/hypernym relations. However, it is well known that Hearst-style patterns occur rarely, such that matching these patterns on the Web in order to alleviate the problem of data sparseness seems a promising solution. In fact, in our case we are not only looking for the hypernym relation (comparable to the *Formal*-role) but for similar patterns conveying a *Constitutive*, *Telic* or *Agentive* relation. Our approach consists of 5 phases; for each *qualia term* (the word we want to find the qualia structure for) we:

1. generate for each qualia role a set of so called *clues*, i.e. search engine queries indicating the relation of interest,
2. download the snippets (abstracts) of the 50 first web search engine results matching the generated clues,
3. part-of-speech-tag the downloaded snippets,
4. match patterns in the form of regular expressions conveying the qualia role of interest, and
5. weight and rank the returned qualia elements according to some measure.

The patterns in our pattern library are actually tuples (p, c) where p is a regular expression defined over part-of-speech tags and c a function $c : string \rightarrow string$ called the *clue*. Given a nominal n and a clue c, the query $c(n)$ is sent to the web search engine and the abstracts of the first m documents matching this query are downloaded. Then the snippets are processed to find matches of the pattern p. For example, given the clue $f(x) =$ "$such\ as\ p(x)''$ and the qualia term *computer* we would download m abstracts matching the query f(computer), i.e. "such as computers". Hereby $p(x)$ is a function returning the plural form of x. We implemented this function as a lookup in a lexicon in which plural nouns are mapped to their base form. With the use of such clues, we thus download a num-

ber of snippets returned by the web search engine in which a corresponding regular expression will probably be matched, thus restricting the linguistic analysis to a few promising pages. The downloaded abstracts are then part-of-speech tagged using QTag (Tufis and Mason, 1998). Then we match the corresponding pattern p in the downloaded snippets thus yielding candidate qualia elements as output. The qualia elements are then ranked according to some measure (compare Section 4), resulting in what we call *Ranked Qualia Structures* (RQSs). The clues and patterns used for the different roles can be found in Tables 1 - 4. In the specification of the clues, the function a(x) returns the appropriate indefinite article – *'a'* or *'an'* – or no article at all for the noun x. The use of an indefinite article or no article at all accounts for the distinction between countable nouns (e.g. such as knife) and mass nouns (e.g. water). The choice between using the articles *'a'*, *'an'* or no article at all is determined by issuing appropriate queries to the web search engine and choosing the article leading to the highest number of results. The corresponding patterns are then matched in the 50 snippets returned by the search engine for each clue, thus leading to up to 50 potential qualia elements per clue and pattern[2]. The patterns are actually defined over part-of-speech tags. We indicate POS-tags in square brackets. However, for the sake of simplicity, we largely omit the POS-tags for the lexical elements in the patterns described in Tables 1 - 4. Note that we use traditional regular expression operators such as $*$ (sequence), $+$ (sequence with at least one element) | (alternative) and ? (option). In general, we define a noun phrase (NP) by the following regular expression: NP:=[DT]? ([JJ])+? [NN(S?)])+[3], where the head is the underlined expression, which is lemmatized and considered as a candidate qualia element. For all the patterns described in this section, the underlined part corresponds to the extracted qualia element. In the patterns for the formal role (compare Table 1), NP_{QT} is a noun phrase with the qualia term as head, whereas NP_F is a noun phrase with the potential qualia element as head. For the constitutive role patterns, we use a noun phrase variant NP' defined by the regular expression NP':= (NP of[IN])? NP (, NP)* ((,)? (and|or) NP)?, which allows to extract enumerations of constituents (compare Table 2). It is important to mention that in the case of expressions such as *"a car comprises a fixed number of basic components"*, *"data mining comprises a range of data analysis techniques"*, *"books consist of a series of dots"*, or *"a conversation is made up of a series of observable interpersonal exchanges"*, only the NP after the preposition 'of' is taken into account as qualia element. The *Telic* Role is in principle acquired in the same way as the *Formal* and *Constitutive* roles with the exception that the qualia element is not only the head of a noun phrase, but also a verb or a verb followed by a noun phrase. Table 3 gives the corresponding clues and patterns. In particular, the returned candidate qualia elements are the lemmatized underlined expressions in PURP:=[VB] NP | NP | be[VBD]. Finally, concerning the clues and patterns for the agentive role shown in Table 4, it is interesting to emphasize the usage of the adjectives 'new' and 'complete'. These adjectives are used in the patterns to increase the expectation for the occurrence of a creation verb. According to our experiments, these patterns are indeed more reliable in finding appropriate qualia elements than the alternative version without the adjectives 'new' and 'complete'. Note that in all patterns, the participle (VBD) is always reduced to base form (VB) via a lexicon lookup. In general, the patterns have been crafted by hand, testing and refining them in an iterative process, paying attention to maximize their coverage but also accuracy. In the future, we plan to exploit an approach to automatically learn the patterns.

Clue	Pattern
Singular	
"a(x) x is a kind of "	NP_{QT} is a kind of NP_F
"a(x) x is"	NP_{QT} is a kind of NP_F
"a(x) x and other"	NP_{QT} (,)? and other NP_F
"a(x) x or other"	NP_{QT} (,)? or other NP_F
Plural	
"such as p(x)"	NP_F such as NP_{QT}
"p(x) and other"	NP_{QT} (,)? and other NP_F
"p(x) or other"	NP_{QT} (,)? or other NP_F
"especially p(x)"	NP_F (,)? especially NP_{QT}
"including p(x)"	NP_F (,)? including NP_{QT}

Table 1: Clues and Patterns for the *Formal* role

[2]For the constitutive role these can be even more due to the fact that we consider enumerations.

[3]Though Qtag uses another part-of-speech tagset, we rely on the well-known Penn Treebank tagset for presentation purposes.

Clue	Pattern
Singular	
"a(x) x is made up of "	NP_{QT} is made up of NP'$_C$
"a(x) x is made of"	NP_{QT} is made of NP'$_C$
"a(x) x comprises"	NP_{QT} comprises (of)? NP'$_C$
"a(x) x consists of"	NP_{QT} consists of NP'$_C$
Plural	
"p(x) are made up of "	NP_{QT} is made up of NP'$_C$
"p(x) are made of"	NP_{QT} are made of NP'$_C$
"p(x) comprise"	NP_{QT} comprise (of)? NP'$_C$
"p(x) consist of"	NP_{QT} consist of NP'$_C$

Table 2: Clues and Patterns for the *Constitutive* Role

Clue	Pattern
Singular	
"purpose of a(x) x is"	purpose of (a|an) x is (to)? PURP
"a(x) is used to"	(a|an) x is used to PURP
Plural	
"purpose of p(x) is"	purpose of p(x) is (to)? PURP
"p(x) are used to"	p(x) are used to PURP

Table 3: Clues and Patterns for the *Telic* Role

4 Ranking Measures

In order to rank the different qualia elements of a given qualia structure, we rely on a certain ranking measure. In our experiments, we analyze four different ranking measures. On the one hand, we explore measures which use the Web to calculate the correlation strength between a qualia term and its qualia elements. These measures are Web-based versions of the Jaccard coefficient (Web-Jac), the Pointwise Mutual Information (Web-PMI) and the conditional probability (Web-P). We also present a version of the conditional probability which does not use the Web but merely relies on the counts of each qualia element as produced by the lexico-syntactic patterns (P-measure). We describe these measures in the following.

4.1 Web-based Jaccard Measure (Web-Jac)

Our web-based Jaccard (Web-Jac) measure relies on the web search engine to calculate the number of documents in which x and y co-occur close to each other, divided by the number of documents each one occurs, i.e.

$$\textit{Web-Jac}(x,y) := \frac{Hits(x * y)}{Hits(x) + Hits(y) - Hits(x\ AND\ y)}$$

So here we are relying on the wildcard operator '*' provided by the Google search engine API[4]. Though the specific function of the '*' operator as implemented by Google is actually unknown, the behavior is similar to the formerly available Altavista NEAR operator[5].

[4]In fact, for the experiments described in this paper we rely on the Google API.

Clue	Pattern
Singular	
"to * a(x) new x"	to [RB]? [VB] a? new x
"to * a(x) complete x"	to [RB]? [VB] a? complete x
"a(x) new has been *"	a? new x has been [VBD]
"a(x) complete x has been *"	a? complete has been [VBD]
Plural	
"to * new p(x)"	to [RB]? [VB] new p(x)
"to * complete p(x)"	to [RB]? [VB] complete p(x)

Table 4: Clues and Patterns for the *Agentive* Role

4.2 Web-based Pointwise Mutual Information (Web-PMI)

In line with Magnini et al. (Magnini et al., 2001), we define a PMI-based measure as follows:

$$Web - PMI(x,y) := log_2 \frac{Hits(x\ AND\ y)\ \textit{MaxPages}}{Hits(y)\ Hits(y)}$$

where maxPages is an approximation for the maximum number of English web pages[6].

4.3 Web-based Conditional Probability (Web-P)

The conditional probability $P(x|y)$ is essentially the probability that x is true given that y is true, i.e.

$$\textit{Web-P}(x,y) := P(x|y) = \frac{P(x,y)}{P(y)} = \frac{Hits(x\ NEAR\ y)}{Hits(y)}$$

whereby $Hits(x\ NEAR\ y)$ is calculated as mentioned above using the '*' operator. In contrast to the measures described above, this one is asymmetric so that order indeed matters. Given a qualia term qt as well as a qualia element qe we actually calculate Web-P(qe,qt) for a specific qualia role.

4.4 Conditional Probability (P)

The non web-based conditional probability essentially differs from the Web-based conditional probability in that we only rely on the qualia elements

[5]Initial experiments indeed showed that counting pages in which the two terms occur near each other in contrast to counting pages in which they merely co-occur improved the results of the Jaccard measure by about 15%.

[6]We determine this number experimentally as the number of web pages containing the words 'the' and 'and'.

matched. On the basis of these, we then calculate the probability of a certain qualia element given a certain role on the basis of its frequency of appearance with respect to the total number of qualia elements derived for this role, i.e. we simply calculate $P(qe|qr, qt)$ on the basis of the derived occurrences, where qt is a given qualia term, qr is the specific qualia role and qe is a qualia element.

5 Evaluation

In this section, we first of all describe our evaluation measures. Then we describe the creation of the gold standard. Further, we present the results of the comparison of the different ranking measures with respect to the gold standard. Finally, we present an *'a posteriori'* evaluation showing that the qualia structures learned are indeed reasonable.

5.1 Evaluation Measures

As our focus is to compare the different measures described above, we need to evaluate their corresponding rankings of the qualia elements for each qualia structure. This is a similar case to evaluating the ranking of documents within information retrieval systems. In fact, as done in standard information retrieval research, our aim is to determine for each ranking the precision/recall trade-off when considering more or less of the items starting from the top of the ranked list. Thus, we evaluate our approach calculating precision at standard recall levels as typically done in information retrieval research (compare (Baeza-Yates and Ribeiro-Neto, 1999)). Hereby the 11 standard recall levels are 0%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% and 100%. Further, precision at these standard recall levels is calculated by interpolating recall as follows: $P(r_j) = max_{r_j \leq r \leq r_{j+1}} P(r)$, where, $j \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. This way we can compare the precision over standard recall figures for the different rankings, thus observing which measure leads to the better precision/recall trade-off.

In addition, in order to provide one single value to compare, we also calculate the F-Measure corresponding to the best precision/recall trade-off for each ranking measure. This F-Measure thus corresponds to the best cut-off point we can find for the items in the ranked list. In fact, we use the well-known F_1 measure corresponding to the harmonic mean between recall and precision:

$$F_1 := max_j \frac{2\ P(r_j)\ r_j}{P(r_j) + r_j}$$

As a baseline, we compare our results to a naive strategy without any ranking, i.e. we calculate the F-Measure for all the items in the (unranked) list of qualia elements. Consequently, for the rankings to be useful, they need to yield higher F-Measures than this naive baseline.

5.2 Gold Standard

The gold standard was created for the 30 words used already in the experiments described in (Yamada and Baldwin, 2004): *accounting, beef, book, car, cash, clinic, complexity, counter, county, delegation, door, estimate, executive, food, gaze, imagination, investigation, juice, knife, letter, maturity, novel, phone, prisoner, profession, review, register, speech, sunshine, table.* These words were distributed more or less uniformly between 30 participants of our experiment, making sure that three qualia structures for each word were created by three different subjects. The participants, who were all non-linguistics, received a short instruction in the form of a short presentation explaining what qualia structures are, the aims of the experiment as well as their specific task. They were also shown some examples for qualia structures for words not considered in our experiments. Further, they were asked to provide between 5 and 10 qualia elements for each qualia role. The participants completed the test via e-mail. As a first interesting observation, it is worth mentioning that the participants only delivered 3-5 qualia elements on average depending on the role in question. This shows already that participants had trouble in finding different qualia elements for a given qualia role. We calculate the agreement for the task of specifying qualia structures for a particular term and role as the averaged pairwise agreement between the qualia elements delivered by the three subjects, henceforth S_1, S_2 and S_3 as:

$$Agr := \frac{\frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} + \frac{|S_1 \cap S_3|}{|S_1 \cup S_3|} + \frac{|S_2 \cup S_3|}{|S_2 \cap S_3|}}{3}$$

Averaging over all the roles and words, we get an average agreement of 11.8%, i.e. our human test

subjects coincide in slightly more than every 10th qualia element. This is certainly a very low agreement and certainly hints at the fact that the task considered is certainly difficult. The agreement was lowest (7.29%) for the telic role.

A further interesting observation is that the lowest agreement is yielded for more abstract words, while the agreement for very concrete words is reasonable. For example, the five words with the highest agreement are indeed concrete things: *knife* (31%), *cash* (29%), *juice* (21%), *car* (20%) and *door* (19%). The words with an agreement below 5% are *gaze*, *prisoner*, *accounting*, *maturity*, *complexity* and *delegation*. In particular, our test subjects had substantial difficulties in finding the purpose of such abstract words. In fact, the agreement on the telic role is below 5% for more than half of the words.

In general, this shows that any automatic approach towards learning qualia structures faces severe limits. For sure, we can not expect the results of an automatic evaluation to be very high. For example, for the telic role of *'clinic'*, one test subject specified the qualia element *'cure'*, while another one specified *'cure disease'*, thus leading to a disagreement in spite of the obvious agreement at the semantic level. In this line, the average agreement reported above has in fact to be regarded as a lower bound for the actual agreement. Of course, our approach to calculating agreement is too strict, but in absence of a clear and computable definition of semantic agreement, it will suffice for the purposes of this paper.

5.3 Gold Standard Evaluation

We ran experiments calculating the qualia structure for each of the 30 words, ranking the resulting qualia elements for each qualia structure using the different measures described in Section 4.

Figure 1 shows the best F-Measure corresponding to a cut-off leading to an optimal precision/recall trade-off. We see that the P-measure performs best, while the Web-P measure and the Web-Jac measure follow at about 0.05 and 0.2 points distance. The PMI-based measure indeed leads to the worst F-Measure values.

Indeed, the P-measure delivered the best results for the formal and agentive roles, while for the constitutive and telic roles the Web-Jac measure performed best. The reason why PMI performs so badly is the fact that it favors too specific results which are unlikely to occur as such in the gold standard. For example, while the conditional probability ranks highest: *explore, help illustrate, illustrate* and *enrich* for the telic role of *novel*, the PMI-based measure ranks highest: *explore great themes*, *illustrate theological points*, *convey truth*, *teach reading skills* and *illustrate concepts*. A series of significance tests (paired Student's t-test at an α-level of 0.05) showed that the three best performing measures (P, Web-P and Web-Jaccard) show no real difference among them, while all three show significant difference to the Web-PMI measure. A second series of significance tests (again paired Student's t-test at an α-level of 0.05) showed that all ranking measures indeed significantly outperform the baseline, which shows that our rankings are indeed reasonable. Interestingly, there seems to be an interesting positive correlation between the F-Measure and the human agreement. For example, for the best performing ranking measure, i.e. the P-measure, we get an average F-Measure of 21% for words with an agreement over 5%, while we get an F-Measure of 9% for words with an agreement below 5%. The reason here probably is that those words and qualia elements for which people are more confident also have a higher frequency of appearance on the Web.

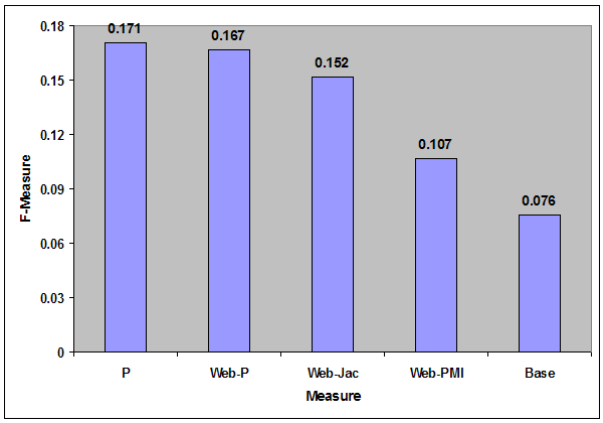


Figure 1: Average F_1 measure for the different ranking measures

5.4 A posteriori Evaluation

In order to check whether the automatically learned qualia structures are reasonable from an intuitive point of view, we also performed an a posteriori

evaluation in the lines of (Cimiano and Wenderoth, 2005). In this experiment, we presented the top 10 ranked qualia elements for each qualia role for 10 randomly selected words to the different test persons. Here we only used the P-measure for ranking as it performed best in our previous evaluation with regard to the gold standard. In order to verify that our sample is not biased, we checked that the F-Measure yielded by our 10 randomly selected words (17.7%) does not differ substantially from the overall average F-Measure (17.1%) to be sure that we have chosen words from all F-Measure ranges. In particular, we asked different test subjects which also participated in the creation of the gold standard to rate the qualia elements with respect to their appropriateness for the qualia term using a scale from 0 to 3, whereby 0 means 'wrong', 1 'not totally wrong', 2 'acceptable' and 3 'totally correct'. The participants confirmed that it was easier to validate existing qualia structures than to create them from scratch, which already corroborates the usefulness of our automatic approach. The qualia structure for each of the 10 randomly selected words was validated independently by three test persons. In fact, in what follows we always report results averaged for three test subjects. Figure 2 shows the average values for different roles. We observe that the constitutive role yields the best results, followed by the formal, telic and agentive roles (in this order). In general, all results are above 2, which shows that the qualia structures produced are indeed acceptable. Though we do not present these results in more detail due to space limitations, it is also interesting to mention that the F-Measure calculated with respect to the gold standard was in general highly correlated with the values assigned by the human test subjects in this *a posteriori* validation.

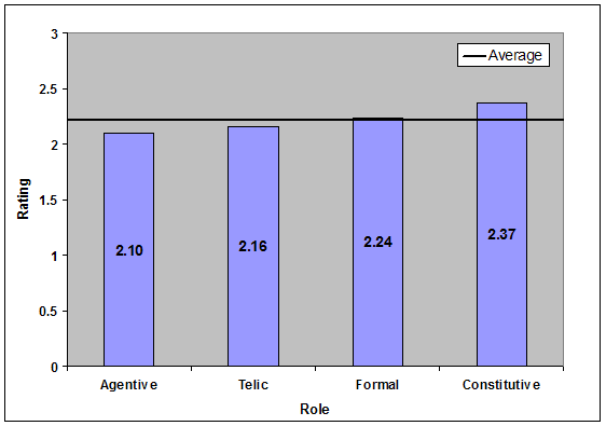


Figure 2: Average ratings for each qualia role

6 Related Work

Instead of matching Hearst-style patterns (Hearst, 1992) in a large text collection, some researchers have recently turned to the Web to match these patterns such as in (Markert et al., 2003) or (Etzioni et al., 2005). Our approach goes further in that it not only learns typing, superconcept or instance-of relations, but also *Constitutive*, *Telic* and *Agentive* relations.

There also exist approaches specifically aiming at learning qualia elements from corpora based on machine learning techniques. Claveau et al. (Claveau et al., 2003) for example use Inductive Logic Programming to learn if a given verb is a qualia element or not. However, their approach does no go as far as learning the complete qualia structure for a lexical element as in our approach. Further, in their approach they do not distinguish between different qualia roles and restrict themselves to verbs as potential fillers of qualia roles.

Yamada and Baldwin (Yamada and Baldwin, 2004) present an approach to learning *Telic* and *Agentive* relations from corpora analyzing two different approaches: one relying on matching certain lexico-syntactic patterns as in the work presented here, but also a second approach consisting in training a maximum entropy model classifier. The patterns used by (Yamada and Baldwin, 2004) differ substantially from the ones used in this paper, which is mainly due to the fact that search engines do not provide support for regular expressions and thus instantiating a pattern as 'V[+ing] Noun' is impossible in our approach as the verbs are unknown a priori.

Poesio and Almuhareb (Poesio and Almuhareb, 2005) present a machine learning based approach to classifying attributes into the six categories: *quality*, *part*, *related-object*, *activity*, *related-agent* and *non-attribute*.

7 Conclusion

We have presented an approach to automatically learning qualia structures from the Web. Such an approach is especially interesting either for lexicog-

raphers aiming at constructing lexicons, but even more for natural language processing systems relying on deep lexical knowledge as represented by qualia structures. In particular, we have focused on learning ranked qualia structures which allow to find an ideal cut-off point to increase the precision/recall trade-off of the learned structures. We have abstracted from the issue of finding the appropriate cut-off, leaving this for future work. In particular, we have evaluated different ranking measures for this purpose, showing that all of the analyzed measures (Web-P, Web-Jaccard, Web-PMI and the conditional probability) significantly outperformed a baseline using no ranking measure. Overall, the plain conditional probability P (not calculated over the Web) as well as the conditional probability calculated over the Web (Web-P) delivered the best results, while the PMI-based ranking measure yielded the worst results. In general, our main aim has been to show that, though the task of automatically learning qualia structures is indeed very difficult as shown by our low human agreement, reasonable structures can indeed be learned with a pattern-based approach as presented in this paper. Further work will aim at inducing the patterns automatically given some seed examples, but also at using the automatically learned structures within NLP applications. The created qualia structure gold standard is available for the community[7].

References

R. Baeza-Yates and B. Ribeiro-Neto. 1999. *Modern Information Retrieval*. Addison-Wesley.

C.F. Baker, C.J. Fillmore, and J.B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of COLING/ACL'98*, pages 86–90.

J. Bos, P. Buitelaar, and M. Mineur. 1995. Bridging as coercive accomodation. In *Working Notes of the Edinburgh Conference on Computational Logic and Natural Language Processing (CLNLP-95)*.

P. Cimiano and J. Wenderoth. 2005. Learning qualia structures from the web. In *Proceedings of the ACL Workshop on Deep Lexical Acquisition*, pages 28–37.

V. Claveau, P. Sebillot, C. Fabre, and P. Bouillon. 2003. Learning semantic lexicons from a part-of-speech and semantically tagged corpus using inductive logic programming. *Journal of Machine Learning Research*, (4):493–525.

[7] See `http://www.cimiano.de/qualia`.

O. Etzioni, M. Cafarella, D. Downey, A-M. Popescu, T. Shaked, S. Soderland, D.S. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134.

C. Fellbaum. 1998. *WordNet, an electronic lexical database*. MIT Press.

M.A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of COLING'92*, pages 539–545.

M. Johnston and F. Busa. 1996. Qualia structure and the compositional interpretation of compounds. In *Proceedings of the ACL SIGLEX workshop on breadth and depth of semantic lexicons*.

A. Kilgariff and G. Grefenstette, editors. 2003. *Special Issue on the Web as Corpus of the Journal of Computational Linguistics*, volume 29(3). MIT Press.

B. Magnini, M. Negri, R. Prevete, and H. Tanev. 2001. Is it the right answer?: exploiting web redundancy for answer validation. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 425–432.

K. Markert, N. Modjeska, and M. Nissim. 2003. Using the web for nominal anaphora resolution. In *Proceedings of the EACL Workshop on the Computational Treatment of Anaphora*.

M. Poesio and A. Almuhareb. 2005. Identifying concept attributes using a classifier. In *Proceedings of the ACL Workshop on Deep Lexical Acquisition*, pages 18–27.

J. Pustejovsky, P. Anick, and S. Bergler. 1993. Lexical semantic techniques for corpus analysis. *Computational Lingustics, Special Issue on Using Large Corpora II*, 19(2):331–358.

J. Pustejovsky. 1991. The generative lexicon. *Computational Linguistics*, 17(4):209–441.

D. Tufis and O. Mason. 1998. Tagging Romanian Texts: a Case Study for QTAG, a Language Independent Probabilistic Tagger. In *Proceedings of LREC*, pages 589–96.

E.M. Voorhees. 1994. Query expansion using lexical-semantic relations. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 61–69.

I. Yamada and T. Baldwin. 2004. Automatic discovery of telic and agentive roles from corpus data. In *Proceedings of the the 18th Pacific Asia Conference on Language, Information and Computation (PACLIC)*.

A Sequencing Model for Situation Entity Classification

Alexis Palmer, Elias Ponvert, Jason Baldridge, and Carlota Smith
Department of Linguistics
University of Texas at Austin
{alexispalmer,ponvert,jbaldrid,carlotasmith}@mail.utexas.edu

Abstract

Situation entities (SEs) are the events, states, generic statements, and embedded facts and propositions introduced to a discourse by clauses of text. We report on the first data-driven models for labeling clauses according to the type of SE they introduce. SE classification is important for discourse mode identification and for tracking the temporal progression of a discourse. We show that (a) linguistically-motivated cooccurrence features and grammatical relation information from deep syntactic analysis improve classification accuracy and (b) using a sequencing model provides improvements over assigning labels based on the utterance alone. We report on genre effects which support the analysis of discourse modes having characteristic distributions and sequences of SEs.

1 Introduction

Understanding discourse requires identifying the participants in the discourse, the situations they participate in, and the various relationships between and among both participants and situations. Coreference resolution, for example, is concerned with understanding the relationships between references to discourse participants. This paper addresses the problem of identifying and classifying references to situations expressed in written English texts.

Situation entities (SEs) are the events, states, generic statements, and embedded facts and propositions which clauses introduce (Vendler, 1967; Verkuyl, 1972; Dowty, 1979; Smith, 1991; Asher, 1993; Carlson and Pelletier, 1995). Consider the text passage below, which introduces an *event*-type entity in (1), a *report*-type entity in (2), and a *state*-type entity in (3).

(1) Sony Corp. has heavily promoted the Video Walkman since the product's introduction last summer ,

(2) but Bob Gerson , video editor of This Week in Consumer Electronics , says

(3) Sony conceives of 8mm as a "family of products , camcorders and VCR decks , "

SE classification is a fundamental component in determining the discourse mode of texts (Smith, 2003) and, along with aspectual classification, for temporal interpretation (Moens and Steedman, 1988). It may be useful for discourse relation projection and discourse parsing.

Though situation entities are well-studied in linguistics, they have received very little computational treatment. This paper presents the first data-driven models for SE classification. Our two main strategies are (a) the use of linguistically-motivated features and (b) the implementation of SE classification as a sequencing task. Our results also provide empirical support for the very notion of discourse modes, as we see clear genre effects in SE classification.

We begin by discussing SEs in more detail. Section 3 describes our two annotated data sets and provides examples of each SE type. Section 4 discusses feature sets, and sections 5 and 6 present models, experiments, and results.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 896–903,
Prague, Czech Republic, June 2007.

2 Discourse modes and situation entities

In this section, we discuss some of the linguistic motivation for SE classification and the relation of SE classification to discourse mode identification.

2.1 Situation entities

The categorization of SEs into aspectual classes is motivated by patterns in their linguistic behavior. We adopt an expanded version of a paradigm relating SEs to discourse mode (Smith, 2003) and characterize SEs with four broad categories:

1. **Eventualities**. *Events* (E), particular *states* (S), and *reports* (R). R is a sub-type of E for SEs introduced by verbs of speech (e.g., *say*).

2. **General statives**. *Generics* (G) and *generalizing sentences* (GS). The former are utterances predicated of a general class or kind rather than of any specific individual. The latter are habitual utterances that refer to ongoing actions or properties predicated of specific individuals.

3. **Abstract entities**. *Facts* (F) and *propositions* (P).[1]

4. **Speech-act types**. *Questions* (Q) and *imperatives* (IMP).

Examples of each SE type are given in section 3.2.

There are a number of linguistic tests for identifying situation entities (Smith, 2003). The term *linguistic test* refers to a rule which correlates an SE type to particular linguistic forms. For example, event-type verbs in simple present tense are a linguistic correlate of GS-type SEs.

These linguistic tests vary in their precision and different tests may predict different SE types for the same clause. A rule-based implementation using them to classify SEs would require careful rule ordering or mediation of rule conflicts. However, since these rules are exactly the sort of information extracted as features in data-driven classifiers, they can be cleanly integrated by assigning them empirically determined weights. We use maximum entropy models (Berger et al., 1996), which are particularly well-suited for tasks (like ours) with many overlapping features, to harness these linguistic insights by using features in our models which encode, directly or indirectly, the linguistic correlates to SE types. The features are described in detail in section 4.

[1] In our system these two SE types are identified largely as complements of factive and propositional verbs as discussed in Peterson (1997). Fact and propositional complements have some linguistic as well as some notional differences. Facts may have causal effects, and facts are in the world. Neither of these is true for propositions. In addition, the two have somewhat different semantic consequences of a presuppositional nature.

2.2 Basic and derived situation types

Situation entities each have a *basic situation type*, determined by the verb plus its arguments, the verb constellation. The verb itself plays a key role in determining basic situation type but it is not the only factor. Changes in the arguments or tense of the verb sometimes change the basic situation types:

(4) Mickey painted the house. (E)

(5) Mickey paints houses. (GS)

If SE type could be determined solely by the verb constellation, automatic classification of SEs would be a relatively straightforward task. However, other parts of the clause often override the basic situation type, resulting in aspectual coercion and a *derived situation type*. For example, a modal adverb can trigger aspectual coercion:

(6) Mickey probably paints houses. (P)

Serious challenges for SE classification arise from the aspectual ambiguity and flexibility of many predicates as well as from aspectual coercion.

2.3 Discourse modes

Much of the motivation of SE classification is toward the broader goal of identifying discourse modes, which provide a linguistic characterization of textual passages according to the situation entities introduced. They correspond to intuitions as to the rhetorical or semantic character of a text. Passages of written text can be classified into modes of discourse – Narrative, Description, Argument, Information, and Report – by examining concrete linguistic cues in the text (Smith, 2003). These cues are of two forms: the distribution of situation entity types and the mode of progression (either temporal or metaphorical) through the text.

For example, the Narration and Report modes both contain mainly events and temporally bounded states; they differ in their principles of temporal progression. Report passages progress with respect to (deictic) speech time, whereas Narrative passages progress with respect to (anaphoric) reference time. Passages in the Description mode are predominantly stative, and Argument mode passages tend to be characterized by propositions and Information mode passages by facts and states.

3 Data

This section describes the data sets used in the experiments, the process for creating annotated training data, and preprocessing steps. Also, we give examples of the ten SE types.

There are no established data sets for SE classification, so we created annotated training data to test our models. We have annotated two data sets, one from the Brown corpus and one based on data from the Message Understanding Conference 6 (MUC6).

3.1 Segmentation

The Brown texts were segmented according to SE-containing clausal boundaries, and each clause was labeled with an SE label. Segmentation is itself a difficult task, and we made some simplifications. In general, clausal complements of verbs like *say* which have clausal direct objects were treated as separate clauses and given an SE label. Clausal complements of verbs which have an entity as a direct object and second clausal complement (such as *notify*) were not treated as separate clauses. In addition, some modifying and adjunct clauses were not assigned separate SE labels.

The MUC texts came to us segmented into elementary discourse units (EDUs), and each EDU was labeled by the annotators. The two data sets were segmented according to slightly different conventions, and we did not normalize the segmentation. The inconsistencies in segmentation introduce some error to the otherwise gold-standard segmentations.

3.2 Annotation

Each text was independently annotated by two experts and reviewed by a third. Each clause was assigned precisely one SE label from the set of ten possible labels. For clauses which introduce more than one SE, the annotators selected the most salient one. This situation arose primarily when complement clauses were not treated as distinct clauses, in which case the SE selected was the one introduced by the main verb. The label N was used for clauses which do not introduce any situation entity.

SE	Text
S	That compares with roughly paperback-book dimensions for VHS.
G	Accordingly, most VHS camcorders are usually bulky and weigh around eight pounds or more.
S	"Carl is a tenacious fellow,"
R	said a source close to USAir.
GS	"He doesn't give up easily
GS	and one should never underestimate what he can or will do."
S	For Jenks knew
F	that Bari's defenses were made of paper.
E	Mr. Icahn then proposed
P	that USAir buy TWA,
IMP	"Fermate"!
R	Musmanno bellowed to his Italian crewmen.
Q	What's her name?
S	Quite seriously, the names mentioned as possibilities were three male apparatchiks from the Beltway's Democratic political machine
N	By Andrew B. Cohen Staff Reporter of The WSJ

Table 1: Example clauses and their SE annotation. Horizontal lines separate extracts from different texts.

The Brown data set consists of 20 "popular lore" texts from section **cf** of the Brown corpus. Segmentation of these texts resulted in a total of 4390 clauses. Of these, 3604 were used for training and development, and 786 were held out as final testing data. The MUC data set consists of 50 Wall Street Journal newspaper articles segmented to a total of 1675 clauses. 137 MUC clauses were held out for testing. The Brown texts are longer than the MUC texts, with an average of 219.5 clauses per document as compared to MUC's average of 33.5 clauses. The average clause in the Brown data contains 12.6 words, slightly longer than the MUC texts' average of 10.9 words.

Table 1 provides examples of the ten SE types as well as showing how clauses were segmented. Each SE-containing example is a sequence of EDUs from the data sets used in this study.

W	
WORDS	words & punctuation
WT	
W	(see above)
POSONLY	POS tag for each word
WORD/POS	word/POS pair for each word
WTL	
WT	(see above)
FORCEPRED	T if clause (or preceding clause) contains *force* predicate
PROPPRED	T if clause (or preceding clause) contains propositional verb
FACTPRED	T if clause (or preceding clause) contains factive verb
GENPRED	T if clause contains generic predicate
HASFIN	T if clause contains finite verb
HASMODAL	T if clause contains modal verb
FREQADV	T if clause contains frequency adverb
MODALADV	T if clause contains modal adverb
VOLADV	T if clause contains volitional adverb
FIRSTVB	lexical item and POS tag for first verb
WTLG	
WTL	(see above)
VERBS	all verbs in clause
VERBTAGS	POS tags for all verbs
MAINVB	main verb of clause
SUBJ	subject of clause (lexical item)
SUPER	CCG supertag

Table 2: Feature sets for SE classification

3.3 Preprocessing

The linguistic tests for SE classification appeal to multiple levels of linguistic information; there are lexical, morphological, syntactic, categorial, and structural tests. In order to access categorial and structural information, we used the C&C[2] toolkit (Clark and Curran, 2004). It provides part-of-speech tags and Combinatory Categorial Grammar (CCG) (Steedman, 2000) categories for words and syntactic dependencies across words.

4 Features

One of our goals in undertaking this study was to explore the use of linguistically-motivated features and deep syntactic features in probabilistic models for SE classification. The nature of the task requires features characterizing the entire clause. Here, we describe our four feature sets, summarized in table 2. The feature sets are additive, extending very basic feature sets first with linguistically-motivated features and then with deep syntactic features.

[2] svn.ask.it.usyd.edu.ap/trac/candc/wiki

4.1 Basic feature sets: W and WT

The WORDS (W) feature set looks only at the words and punctuation in the clause. These features are obtained with no linguistic processing.

WORDS/TAGS (WT) incorporates part-of-speech (POS) tags for each word, number, and punctuation mark in the clause and the word/tag pairs for each element of the clause. POS tags provide valuable information about syntactic category as well as certain kinds of shallow semantic information (such as verb tense). The tags are useful for identifying verbs, nouns, and adverbs, and the words themselves represent lexico-semantic information in the feature sets.

4.2 Linguistically-motivated feature set: WTL

The WORDS/TAGS/LINGUISTIC CORRELATES (WTL) feature set introduces linguistically-motivated features gleaned from the literature on SEs; each feature encodes a linguistic cue that may correlate to one or more SE types. These features are not directly annotated; instead they are extracted by comparing words and their tags for the current and immediately preceding clauses to lists containing appropriate triggers. The lists are compiled from the literature on SEs.

For example, clauses embedded under predicates like *force* generally introduce E-type SEs:

(7) I forced [John to run the race with me].

(8) * I forced [John to know French].

The feature `force-PREV` is extracted if a member of the *force*-type predicate word list occurs in the previous clause.

Some of the correlations discussed in the literature rely on a level of syntactic analysis not available in the WTL feature set. For example, stativity of the main verb is one feature used to distinguish between event and state SEs, and particular verbs and verb tenses have tendencies with respect to stativity. To approximate the main verb without syntactic analysis, WTL uses the lexical item of the first verb in the clause and the POS tags of all verbs in the clause.

These linguistic tests are non-absolute, making them inappropriate for a rule-based model. Our models handle the defeasibility of these correlations probabilistically, as is standard for machine learning for natural language processing.

4.3 Addition of deep features: WTLG

The WORDS/TAGS/LINGUISTIC CORRELATES/GRAMMATICAL RELATIONS (WTLG) feature set uses a deeper level of syntactic analysis via features extracted from CCG parse representations for each clause. This feature set requires an additional step of linguistic processing but provides a basis for more accurate classification.

WTL approximated the main verb by sloppily taking the first verb in the clause; in contrast, WTLG uses the main verb identified by the parser. The parser also reliably identifies the subject, which is used as a feature.

Supertags –CCG categories assigned to words– provide an interesting class of features in WTLG. They succinctly encode richer grammatical information than simple POS tags, especially subcategorization and argument types. For example, the tag S\NP denotes an intransitive verb, whereas (S\NP)/NP denotes a transitive verb. As such, they can be seen as a way of encoding the verbal constellation and its effect on aspectual classification.

5 Models

We consider two types of models for the automatic classification of situation entities. The first, a labeling model, utilizes a maximum entropy model to predict SE labels based on clause-level linguistic features as discussed above. This model ignores the discourse patterns that link multiple utterances. Because these patterns recur, a sequencing model may be better suited to the SE classification task. Our second model thus extends the first by incorporating the previous n ($0 \leq n \leq 6$) labels as features.

Sequencing is standardly used for tasks like part-of-speech tagging, which generally assume smaller units to be both tagged and considered as context for tagging. We are tagging at the clause level rather than at the word level, but the structure of the problem is essentially the same. We thus adapted the OpenNLP maximum entropy part-of-speech tagger[3] (Hockenmaier et al., 2004) to extract features from utterances and to tag sequences of utterances instead of words. This allows the use of features of adjacent clauses as well as previously-predicted labels when making classification decisions.

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

6 Experiments

In this section we give results for testing on Brown data. All results are reported in terms of accuracy, defined as the percentage of correctly-labeled clauses. Standard 10-fold cross-validation on the training data was used to develop models and feature sets. The optimized models were then tested on the held-out Brown and MUC data.

The baseline was determined by assigning *S* (state), the most frequent label in both training sets, to each clause. Baseline accuracy was 38.5% and 36.2% for Brown and MUC, respectively.

In general, accuracy figures for MUC are much higher than for Brown. This is likely due to the fact that the MUC texts are more consistent: they are all newswire texts of a fairly consistent tone and genre. The Brown texts, in contrast, are from the 'popular lore' section of the corpus and span a wide range of topics and text types. Nonetheless, the patterns between the feature sets and use of sequence prediction hold across both data sets; here, we focus our discussion on the results for the Brown data.

6.1 Labeling results

The results for the labeling model appear in the two columns labeled 'n=0' in table 3. On Brown, the simple W feature set beats the baseline by 6.9% with an accuracy of 45.4%. Adding POS information (WT) boosts accuracy 4.5% to 49.9%. We did not see the expected increase in performance from the linguistically motivated WTL features, but rather a slight decrease in accuracy to 48.9%. These features may require a greater amount of training material to be effective. Addition of deep linguistic information with WTLG improved performance to 50.6%, a gain of 5.2% over words alone.

6.2 Oracle results

To determine the potential effectiveness of sequence prediction, we performed oracle experiments on Brown by including previous gold-standard labels as features. Figure 1 illustrates the results from oracle experiments incorporating from zero to six previous gold-standard SE labels (the *lookback*). The increase in performance illustrates the importance of context in the identification of SEs and motivates the use of sequence prediction.

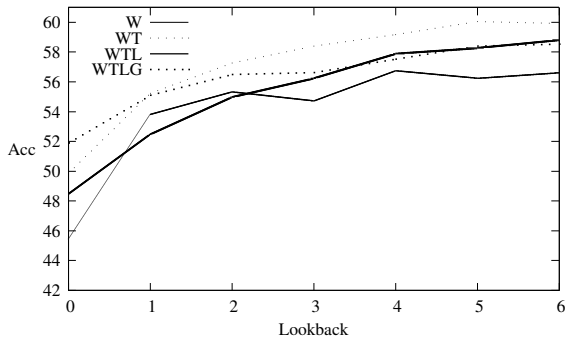


Figure 1: **Oracle results** on Brown data.

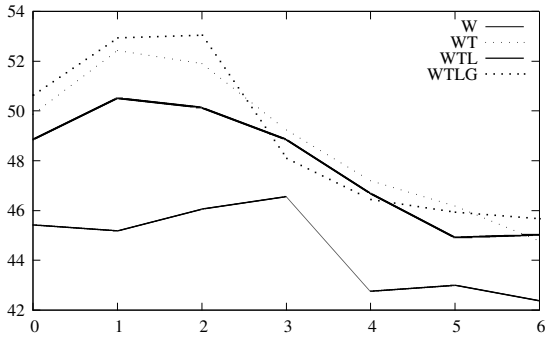


Figure 2: Sequencing results on Brown data.

6.3 Sequencing results

Table 3 gives the results of classification with the sequencing model on the Brown data. As with the labeling model, accuracy is boosted by WT and WTLG feature sets. We see an unexpected degradation in performance in the transition from WT to WTL.

The most interesting results here, though, are the gains in accuracy from use of previously-predicted labels as features for classification. When labeling performance is relatively poor, as with feature set W, previous labels help very little, but as labeling accuracy increases, previous labels begin to effect noticeable increases in accuracy. For the best two feature sets, considering the previous two labels raises the accuracy 2.0% and 2.5%, respectively.

In most cases, though, performance starts to degrade as the model incorporates more than two previous labels. This degradation is illustrated in Figure 2. The explanation for this is that the model is still very weak, with an accuracy of less than 54% for the Brown data. The more previous predicted labels the model conditions on, the greater the likelihood that one or more of the labels is incorrect. With gold-standard labels, we see a steady increase in accuracy as we look further back, and we would need a better performing model to fully take advantage of knowledge of SE patterns in discourse.

The sequencing model plays a crucial role, particularly with such a small amount of training material, and our results indicate the importance of local context in discourse analysis.

BROWN	Lookback (n)						
	0	1	2	3	4	5	6
W	45.4	45.2	46.1	46.6	42.8	43.0	42.4
WT	49.9	52.4	51.9	49.2	47.2	46.2	44.8
WTL	48.9	50.5	50.1	48.9	46.7	44.9	45.0
WTLG	50.6	52.9	**53.1**	48.1	46.4	45.9	45.7
Baseline	38.5						

Table 3: SE classification results with sequencing on **Brown test set**. Bold cell indicates accuracy attained by model parameters that performed best on development data.

6.4 Error analysis

Given that a single one of the ten possible labels occurs for more than 35% of clauses in both data sets, it is useful to look at the distribution of errors over the labels. Table 4 is a confusion matrix for the held-out Brown data using the best feature set.[4] The first column gives the label and number of occurrences of that label, and the second column is the accuracy achieved for that label. The next two columns show the percentage of erroneous labels taken by the labels S and GS. These two labels are the most common labels in the development set (38.5% and 32.5%). The final column sums the percentages of errors assigned to the remaining seven labels. As one would expect, the model learns the predominance of these two labels. There are a few interesting points to make about this data.

First, 66% of G-type clauses are mistakenly assigned the label GS. This is interesting because these two SE-types constitute the broader SE cat-

[4]Thanks to the anonymous reviewer who suggested this useful way of looking at the data.

Label	% Correct Label	% Incorrect S	GS	Other
S(278)	72.7	n/a	14.0	13.3
E(203)	50.7	37.0	11.8	0.5
GS(203)	44.8	46.3	n/a	8.9
R(26)	38.5	30.8	11.5	19.2
N(47)	23.4	31.9	23.4	21.3
G(12)	0.0	25.0	66.7	8.3
IMP(8)	0.0	75.0	25.0	0.0
P(7)	0.0	71.4	28.6	0.0
F(2)	0.0	100.0	0.0	0.0

Table 4: Confusion matrix for Brown held-out test data, WTLG feature set, lookback n = 2. Numbers in parentheses indicate how many clauses have the associated gold standard label.

egory of generalizing statives. The distribution of errors for R-type clauses points out another interesting classification difficulty.[5] Unlike the other categories, the percentage of *false-other* labels for R-type clauses is higher than that of *false-GS* labels. 80% of these *false-other* labels are of type E. The explanation for this is that R-type clauses are a subtype of the event class.

6.5 Genre effects in classification

Different text domains frequently have different characteristic properties. Discourse modes are one way of analyzing these differences. It is thus interesting to compare SE classification when training and testing material come from different domains.

Table 5 shows the performance on Brown when training on Brown and/or MUC using the WTLG feature set with simple labeling and with sequence prediction with a lookback of two. A number of things are suggested by these figures. First, the labeling model (lookback of zero), beats the baseline even when training on out-of-domain texts (43.1% vs. 38.5%), but this is unsurprisingly far below training on in-domain texts (43.1% vs. 50.6%). Second, while sequence prediction helps with in-domain training (53.1% vs 50.6%), it makes no difference with out-of-domain training (42.9% vs 43.1%). This indicates that the patterns of SEs in a text do indeed correlate with domains and their discourse modes, in line with case-studies in the discourse modes theory (Smith, 2003). Finally, mixing out-of-domain training material with in-domain material does not hurt labelling accuracy (50.4% vs 50.6%), but it does take away the gains from sequencing (49.5% vs 53.1%).

[5]Thanks to an anonymous reviewer for bringing this to our attention.

WTLG	**lookback**	Brown test set
train:Brown	0	50.6
	2	53.1
train:MUC	0	43.1
	2	42.9
train:all	0	50.4
	2	49.5

Table 5: Cross-domain SE classification

These genre effects are suggestive, but inconclusive. A similar setup with much larger training and testing sets would be necessary to provide a clearer picture of the effect of mixed domain training.

7 Related work

Though we are aware of no previous work in SE classification, others have focused on automatic detection of aspectual and temporal data.

Klavans and Chodorow (1992) laid the foundation for probabilistic verb classification with their interpretation of aspectual properties as gradient and their use of statistics to model the gradience. They implement a single linguistic test for stativity, treating lexical properties of verbs as tendencies rather than absolute characteristics.

Linguistic indicators for aspectual classification are also used by Siegel (1999), who evaluates 14 indicators to test verbs for stativity and telicity. Many of his indicators overlap with our features.

Siegel and McKeown (2001) address classification of verbs for stativity (event vs. state) and for completedness (culminated vs. non-culminated events). They compare three supervised and one unsupervised machine learning systems. The systems obtain relatively high accuracy figures, but they are domain-specific, require extensive human supervision, and do not address aspectual coercion.

Merlo and Stevenson (2001) use corpus-based thematic role information to identify and classify unergative, unaccusative, and object-drop verbs. Stevenson and Merlo note that statistical analysis cannot and should not be separated from deeper linguistic analysis, and our results support that claim.

The advantages of our approach are the broadened conception of the classification task and the use of sequence prediction to capture a wider context.

8 Conclusions

Situation entity classification is a little-studied but important classification task for the analysis of discourse. We have presented the first data-driven models for SE classification, motivating the treatment of SE classification as a sequencing task.

We have shown that linguistic correlations to situation entity type are useful features for probabilistic models, as are grammatical relations and CCG supertags derived from syntactic analysis of clauses. Models for the task perform poorly given very basic feature sets, but minimal linguistic processing in the form of part-of-speech tagging improves performance even on small data sets used for this study. Performance improves even more when we move beyond simple feature sets and incorporate linguistically-motivated features and grammatical relations from deep syntactic analysis. Finally, using sequence prediction by adapting a POS-tagger further improves results.

The tagger we adapted uses beam search; this allows tractable use of maximum entropy for each labeling decision but forgoes the ability to find the *optimal* label sequence using dynamic programming techniques. In contrast, Conditional Random Fields (CRFs) (Lafferty et al., 2001) allow the use of maximum entropy to set feature weights with efficient recovery of the optimal sequence. Though CRFs are more computationally intensive, the small set of SE labels should make the task tractable for CRFs.

In future, we intend to test the utility of SEs in discourse parsing, discourse mode identification, and discourse relation projection.

Acknowledgments

This work was supported by the Morris Memorial Trust Grant from the New York Community Trust. The authors would like to thank Nicholas Asher, Pascal Denis, Katrin Erk, Garrett Heifrin, Julie Hunter, Jonas Kuhn, Ray Mooney, Brian Reese, and the anonymous reviewers.

References

N. Asher. 1993. *Reference to Abstract objects in Discourse*. Kluwer Academic Publishers.

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

G. Carlson and F. J. Pelletier, editors. 1995. *The Generic Book*. University of Chicago Press, Chicago.

S. Clark and J. R. Curran. 2004. Parsing the WSJ using CCG and log–linear models. In *Proceedings of ACL–04*, pages 104–111, Barcelona, Spain.

D. Dowty. 1979. *Word Meaning and Montague Grammar*. Reidel, Dordrecht.

J. Hockenmaier, G. Bierner, and J. Baldridge. 2004. Extending the coverage of a CCG system. *Research on Language and Computation*, 2:165–208.

J. L. Klavans and M. S. Chodorow. 1992. Degrees of stativity: The lexical representation of verb aspect. In *Proceedings of COLING 14*, Nantes, France.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labelling sequence data. In *Proceedings of ICML*, pages 282–289, Williamstown, USA.

P. Merlo and S. Stevenson. 2001. Automatic verb classification based on statistical distributions of argument structure. *Computational Linguistics*.

M. Moens and M. Steedman. 1988. Temporal ontology and temporal reference. *Computational Linguistics*, 14(2):15–28.

P. Peterson. 1997. *Fact Proposition Event*. Kluwer.

E. V. Siegel and K. R. McKeown. 2001. Learning methods to combine linguistic indicators: Improving aspectual classification and revealing linguistic insights. *Computational Linguistics*, 26(4):595–628.

E. V. Siegel. 1999. Corpus-based linguistic indicators for aspectual classification. In *Proceedings of ACL37*, University of Maryland, College Park.

C. S. Smith. 1991. *The Parameter of Aspect*. Kluwer.

C. S. Smith. 2003. *Modes of Discourse*. Cambridge University Press.

M. Steedman. 2000. *The Syntactic Process*. MIT Press/Bradford Books.

Z. Vendler, 1967. *Linguistics in Philosophy*, chapter Verbs and Times, pages 97–121. Cornell University Press, Ithaca, New York.

H. Verkuyl. 1972. *On the Compositional Nature of the Aspects*. Reidel, Dordrecht.

Words and Echoes: Assessing and Mitigating the Non-Randomness Problem in Word Frequency Distribution Modeling

Marco Baroni
CIMeC (University of Trento)
C.so Bettini 31
38068 Rovereto, Italy
marco.baroni@unitn.it

Stefan Evert
IKW (University of Osnabrück)
Albrechtstr. 28
49069 Osnabrück, Germany
stefan.evert@uos.de

Abstract

Frequency distribution models tuned to words and other linguistic events can predict the number of distinct types and their frequency distribution in samples of arbitrary sizes. We conduct, for the first time, a rigorous evaluation of these models based on cross-validation and separation of training and test data. Our experiments reveal that the prediction accuracy of the models is marred by serious overfitting problems, due to violations of the random sampling assumption in corpus data. We then propose a simple pre-processing method to alleviate such non-randomness problems. Further evaluation confirms the effectiveness of the method, which compares favourably to more complex correction techniques.

1 Introduction

Large-Number-of-Rare-Events (LNRE) models (Baayen, 2001) are a class of specialized statistical models that allow us to estimate the characteristics of the distribution of type probabilities in type-rich linguistic populations (such as words) from limited samples (our corpora). They also allow us to extrapolate quantities such as vocabulary size (the number of distinct types) and the number of hapaxes (types occurring just once) beyond a given corpus or make predictions for completely unseen data from the same underlying population.

LNRE models have applications in theoretical linguistics, e.g. for comparing the type richness of morphological or syntactic processes that are attested to different degrees in the data (Baayen, 1992). Consider for example a very common prefix such as *re-* and a rather rare prefix such as *meta-*. With LNRE models we can answer questions such as: If we could obtain as many tokens of *meta-* as we have of *re-*, would we also see as many distinct types? In other words, is the prefix *meta-* as productive as the prefix *re-*? Practical NLP applications, on the other hand, include estimating how many out-of-vocabulary words we will encounter given a lexicon of a certain size, or making informed guesses about type counts in very large data sets (e.g., *how many typos are there on the Internet?*)

In this paper, after introducing LNRE models (Section 2), we present an evaluation of their performance based on separate training and test data as well as cross-validation (Section 3). As far as we know, this is the first time that such a rigorous evaluation has been conducted. The results show how evaluating on the training set, a common strategy in LNRE research, favours models that overfit the training data and perform poorly on unseen data. They also confirm the observation by Evert and Baroni (2006) that current LNRE models achieve only unsatisfactory prediction accuracy, and this is the issue we turn to in the second part of the paper (Section 4). Having identified the violation of the random sampling assumption by real-world data as one of the main factors affecting the quality of the models, we present a new approach to alleviating non-randomness problems. Further evaluation shows our solution to outperform Baayen's (2001) partition-adjustment method, the former state-of-the-art in non-randomness correction. Section 5 concludes by

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 904–911,
Prague, Czech Republic, June 2007.

pointing out directions for future work.

2 LNRE models

Baayen (2001) introduces a family of models for Zipf-like frequency distributions of linguistic populations, referred to as *LNRE models*. Such a linguistic population is formally described by a finite or countably infinite set of types ω_i and their occurrence probabilities π_i. Word frequency models are not concerned with the probabilities (i.e., relative frequencies) of specific individual types, but rather the overall distribution of these probabilities.

Numbering the types in order of decreasing probability ($\pi_1 \geq \pi_2 \geq \pi_3 \geq \ldots$, called a population *Zipf ranking*), we can specify a LNRE model for their distribution as a function that computes π_i from the Zipf rank i of ω_i. For instance, the Zipf-Mandelbrot law[1] is defined by the equation

$$\pi_i = \frac{C}{(i+b)^a} \tag{1}$$

with parameters $a > 1$ and $b > 0$. It is mathematically more convenient to formulate LNRE models in terms of a type density function $g(\pi)$ on the interval $\pi \in [0, 1]$, such that

$$\int_A^B g(\pi)\, d\pi \tag{2}$$

is the (approximate) number of types ω_i with $A \leq \pi_i \leq B$. Evert (2004) shows that Zipf-Mandelbrot corresponds to a type density of the form

$$g(\pi) := \begin{cases} C \cdot \pi^{-\alpha-1} & A \leq \pi \leq B \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

with parameters $0 < \alpha < 1$ and $0 \leq A < B$.[2] Models that are formulated in terms of such a type density g have many direct applications (e.g. using g as a Bayesian prior), and we refer to them as *proper* LNRE models.

Assuming that a corpus of N tokens is a random sample from such a population, we can make predictions about lexical statistics such as the number $V(N)$ of different types in the corpus (the *vocabulary size*), the number $V_1(N)$ of *hapax legomena* (types occurring just once), as well as the further distribution of type frequencies $V_m(N)$. Since the precise values would be different from sample to sample, the model predictions are given by expectations $E[V(N)]$ and $E[V_m(N)]$, which can be computed with relative ease from the type density function g.

By comparing expected and observed values of V and V_m (for the lowest frequency ranks, usually up to $m = 15$), the parameters of a LNRE model can be estimated (we refer to this as *training* the model), allowing inferences about the population (such as the total number of types in the population) as well as further applications of the estimated type density (e.g. for Good-Turing smoothing). Since we can calculate expected values for samples of arbitrary size N, we can use the trained model to predict how many new types would be seen in a larger corpus, how many hapaxes there would be, etc. This kind of vocabulary growth *extrapolation* has become one of the most important applications of LNRE models in linguistics and NLP.

A detailed account of the mathematics of LNRE models can be found in Baayen (2001, Ch. 2). Baayen describes two LNRE models, *lognormal* and *GIGP*, as well as several other approaches (including a version of *Zipf's law* and the *Yule-Simon* model) that are not based on a type density and hence do not qualify as proper LNRE models. Two LNRE models based on Zipf's law, *ZM* and *fZM*, are introduced by Evert (2004).

In the following, we will only consider proper LNRE models because of their considerably greater utility, and because their performance in extrapolation tasks appears to be better than, or at least comparable to, the other models (Evert and Baroni, 2006). In addition, we exclude the lognormal model because of its computational complexity and numerical instability.[3] In initial evaluation experiments, the performance of lognormal was also inferior to the remaining three models (ZM, fZM and GIGP). Note that ZM is the most simplistic model, with only 2 parameters and assuming an infinite population vocabulary, while fZM and GIGP have 3 parameters

[1] The Zipf-Mandelbrot law is an extension of Zipf's law (which has $a = 1$ and $b = 0$). While the latter originally refers to type frequencies in a given sample, the Zipf-Mandelbrot law is formulated for type probabilities in a population.

[2] In this equation, C is a normalizing constant required in order to ensure $\int_0^1 \pi g(\pi)\, d\pi = 1$, the equivalent of $\sum_i \pi_i = 1$.

[3] There are no closed form equations for the expectations of the lognormal model, which have to be calculated by numerical integration.

and can model populations of different sizes.

3 Evaluation of LNRE models

LNRE models are traditionally evaluated by looking at how well expected values generated by them fit empirical counts extracted from the same dataset used for parameter estimation, often by visual inspection of differences between observed and predicted data in plots. More rigorously, Baayen (2001) and Evert (2004) compare the frequency distribution observed in the training set to the one predicted by the model with a multivariate chi-squared test. As we will show below, evaluating standard LNRE models on the same data that were used to estimate their parameters favours overfitting, which results in poor performance on unseen data.

Evert and Baroni (2006) attempt, for the first time, to evaluate LNRE models on unseen data. However, rather than splitting the data into separate training and test sets, they evaluate the models in an extrapolation setting, where the parameters of the model are estimated on a *subset* of the data used for testing. Evert and Baroni do not attempt to cross-validate the results, and they do not provide a quantitative evaluation, relying instead on visual inspection of empirical and observed vocabulary growth curves.

3.1 Data and procedure

We ran our experiments with three corpora in different languages and representing different textual typologies: the British National Corpus (BNC), a "balanced" corpus of British English of about 100 million tokens illustrating different communicative settings, genres and topics; the deWaC corpus, a Web-crawled corpus of about 1.5 billion German words; and the la Repubblica corpus, an Italian newspaper corpus of about 380 million words.[4]

From each corpus, we extracted 20 non-overlapping samples of randomly selected documents, amounting to a total of 4 million tokens each (punctuation marks and entirely non-alphabetical tokens were removed before sampling, and all words were converted to lowercase). Each of these samples was then split into a training set of 1 million tokens (the *training size* N_0) and a test set of 3 million tokens. The documents in the la Repubblica samples were ordered chronologically before splitting, to simulate a typical scenario arising when working with newspaper data, where the data available for training precede, chronologically, the data one wants to generalize to.

We estimate parameters of the ZM, fZM and GIGP models on each training set, using the zipfR toolkit.[5] The models are then used to predict the expected number of distinct types, i.e., vocabulary size V, at sample sizes of 1, 2 and 3 million tokens, equivalent to 1, 2 and 3 times the size of the training set (we refer to these as the *prediction sizes* N_0, $2N_0$ and $3N_0$, respectively). Finally, the expected vocabulary size $E[V(N)]$ is compared to the observed value $V(N)$ in the test set for $N = N_0$, $N = 2N_0$ and $N = 3N_0$. We also look at $V_1(N)$, the number of hapax legomena, in the same way.

Our main focus is V prediction, since this is by far the most useful measure in practical applications, where we are typically interested in knowing how many types (or how many types belonging to a certain category) we will see as our sample size increases (How many typos are there on the Web? How many types with prefix *meta-* would we see if we had as many types of *meta-* as we have of *re-*?) Hapax legomena counts, on the other hand, play a central role in quantifying morphological productivity (Baayen, 1992) and they give us a first insight into how good the models are at predicting frequency distributions, besides vocabulary size (as we will see, a model's success in predicting V does not necessary imply that the model is also capturing the right frequency distribution).

For all models, corpora and prediction sizes, goodness-of-fit of the model on the training set is measured with a multivariate chi-squared test (Baayen, 2001, 118-122). Performance of the models in prediction of V is assessed via *relative error*, computed for each of the 20 samples from a corpus and the 3 prediction sizes as follows:

$$e = \frac{E[V(N)] - V(N)}{V(N)}$$

where $N = k \cdot N_0$ is the prediction size (for $k = 1, 2, 3$), $V(N)$ is the observed V in the relevant test

[4] See `www.natcorp.ox.ac.uk`, `http://wacky.sslmit.unibo.it` and `http://sslmit.unibo.it/repubblica`

[5] `http://purl.org/stefan.evert/zipfR`

set at size N, and $E[V(N)]$ is the corresponding expected V predicted by a model.[6]

For each corpus and prediction size we obtain 20 values e_i (viz., $e_1, \ldots, e_{20}$). As a summary measure, we report the square root of the mean square relative error (rMSE) calculated according to

$$\sqrt{\text{rMSE}} = \sqrt{\frac{1}{20} \cdot \sum_{i=1}^{20} (e_i)^2}$$

This gives us an overall assessment of prediction accuracy (we take the square root to obtain values on the same scale as relative errors, and thus easier to interpret). We complement rMSEs with reports on the average relative error (indicating whether there is a systematic under- or overestimation bias) and its asymptotic 95% confidence intervals, based on the empirical standard deviation of the e_i across the 20 trials (the confidence intervals are usually somewhat larger than the actual range of values found in the experiments, so they should be seen as "pessimistic estimates" of the actual variance).

3.2 Results

The panels of Figure 1 report rMSE values for the 3 corpora and for each prediction size. For now, we focus on the first 3 histograms of each panel, that present rMSEs for the 3 LNRE models introduced above: ZM, fZM and GIGP (the remaining histograms will be discussed later).[7]

For all corpora and all extrapolation sizes beyond N_0, the simple ZM model outperforms the more sophisticated fZM and GIGP models (which seem to be very similar to each other). Even at the largest prediction size of $3N_0$, ZM's rMSE is well below 10%, whereas the other models have, in the worst case (BNC $3N_0$), a rMSE above 15%. Figure 2 presents plots of average relative error and its empirical confidence intervals (again, focus for now on the ZM, fZM and GIGP results; the rest of the figure is discussed later). We see that the poor performance of fZM and GIGP is due to their tendency to underestimate the true vocabulary size V, while variance is comparable across models.

The rMSEs of V_1 prediction are reported in Figure 3. V_1 prediction performance is poorer across the board, and ZM is no longer outperforming the other models. For space reasons, we do not present relative error and variance plots for V_1, but the general trends are the same observed for V, except that the bias of ZM towards V_1 overestimation is much clearer than for V.

Interestingly, goodness-of-fit on the training data is not a good predictor of V and V_1 prediction performance on unseen data. This is shown in Figure 4, which plots rMSE for prediction of V against goodness-of-fit (quantified by multivariate X^2 on the training set, as discussed above) for all corpora and LNRE models at the $3N_0$ prediction size (but the same patterns emerge at other prediction sizes and with V_1). The larger X^2, the poorer the training set fit; the larger rMSE, the worse the prediction. Thus, ideally, we should see a positive correlation between X^2 and rMSE. Focusing for now on the circles (pinpointing the ZM, fZM and GIGP models), we see that there is instead a *negative* correlation between goodness of fit on the training set and quality of prediction on unseen data.[8]

First, these results indicate that, if we take goodness of fit on the training set as a criterion for choosing the best model (as done by Baayen and Evert), we end up selecting the *worst* model for actual prediction tasks. This is, we believe, a very strong case for applying the split train-test cross-validation method used in other areas of statistical NLP to frequency distribution modeling. Second, the data suggest that the more sophisticated models are *overfitting* the training set, leading to poorer performance than the simpler ZM on unseen data. We turn now to what we think is the main cause for this overfitting.

4 Non-randomness and echoes

The results in the previous section indicate that the Vs predicted by LNRE models are at best "ballpark estimates" (and V_1 predictions, with a relative error that is often above 20%, do not even qualify as plau-

[6] We normalize by $V(N)$ rather than (a function of) $E[V(N)]$ because in the latter case we would favour models that overestimate V, compared to ones that are equally "close" to the correct value but underestimate V.

[7] A table with the full numerical results is available upon request; we find, however, that graphical summaries such as those presented in this paper make the results easier to interpret.

[8] With correlation coefficients of $r < -.8$, significant at the 0.01 level despite the small sample size.

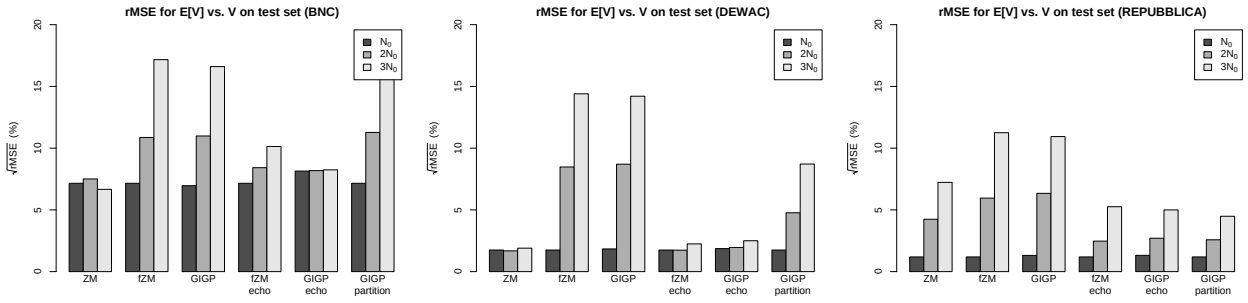


Figure 1: rMSEs of predicted V on the BNC, deWaC and la Repubblica data-sets

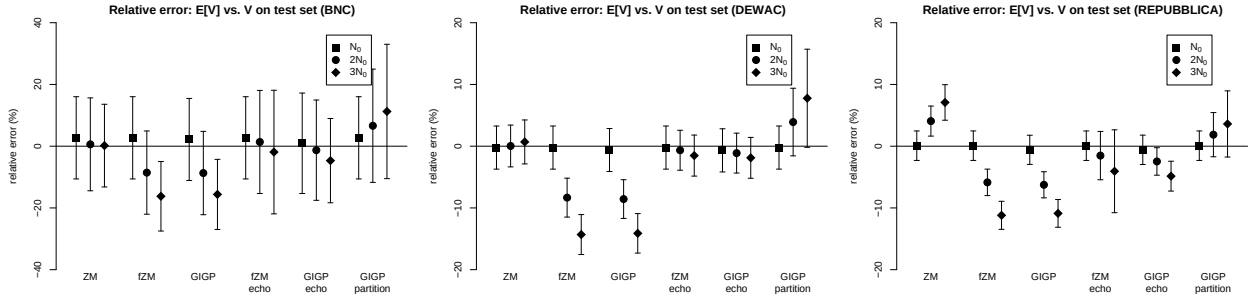


Figure 2: Average relative errors and asymptotic 95% confidence intervals of V prediction on BNC, deWaC and la Repubblica data-sets

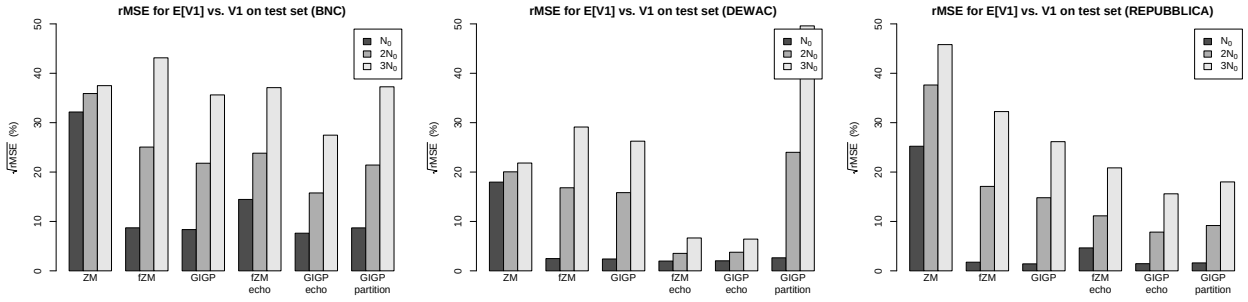


Figure 3: rMSEs of predicted V_1 on the BNC, deWaC and la Repubblica data-sets

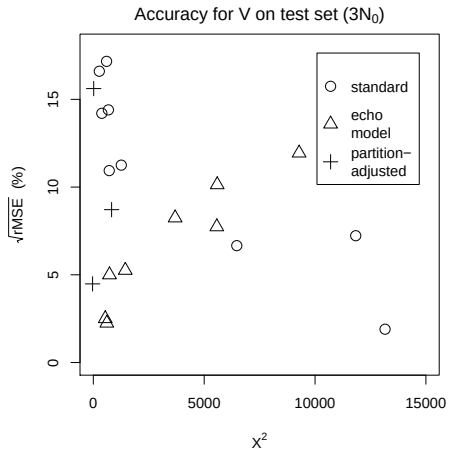


Figure 4: Correlation between X^2 and V prediction rMSE across corpora and models

sible ballpark estimates). Although such rough estimates might be more than adequate for many practical applications, is it possible to further improve the quality of LNRE predictions?

A major factor hampering prediction quality is that real texts massively violate the *randomness assumption* made in LNRE modeling: words, rather obviously, are not picked at random on the basis of their population probability (Evert and Baroni, 2006; Baayen, 2001). The topic-driven "clumpiness" of low frequency content words reduces the number of hapax legomena and other rare events used to estimate the parameters of LNRE models, leading the models to underestimate the type richness of the population. Interestingly (but unsurprisingly), ZM with its assumption of an infinite population, is less prone to this effect, and thus it has a better prediction performance than the more sophisticated fZM and GIGP models, despite its poor goodness-of-fit.

The effect of non-randomness is illustrated very clearly for the BNC (but the same could be shown for the other corpora) by Figure 5, a comparison of rMSE for prediction of V from our experiments above to results obtained on versions of the BNC samples with words scrambled in random order, thus forcibly removing non-randomness effects. We see from this figure that the performance of both fZM and GIGP improves dramatically when they are trained and tested on randomized sequences of words. Interestingly, randomization has instead a *negative* effect on ZM performance.

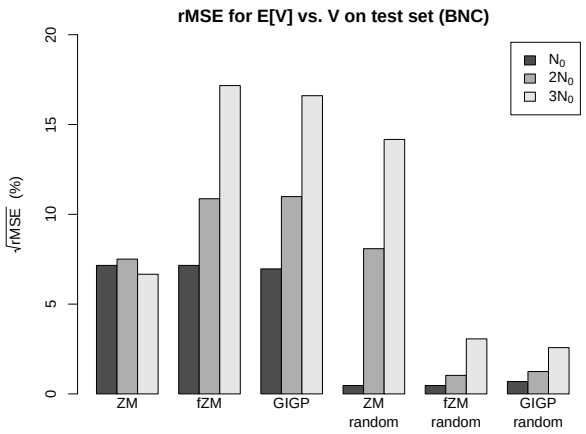


Figure 5: rMSEs of predicted V on unmodified vs. randomized versions of the BNC sets

4.1 Previous approaches to non-randomness

While non-randomness is widely acknowledged as a serious problem for the statistical analysis of corpus data, very few authors have suggested correction strategies. The key problem of non-random data seems to be that the occurrence frequencies of a type in different documents do not follow the binomial distribution assumed by random sampling models. One approach is therefore to model this distribution explicitly, replacing the binomial with its single parameter π by a more complex distribution that has additional parameters (Church and Gale, 1995; Katz, 1996). However, these distributions are currently not applicable to LNRE modeling, which is based on the overall frequencies of types in a corpus rather than their frequencies in individual documents. The overall frequencies can only be calculated by summation over all documents in the corpus, resulting in a mathematically and numerically intractable model. In addition, the type density $g(\pi)$ would have to be extended to a multi-dimensional function, requiring a large number of parameters to be estimated from the data.

Baayen (2001) suggests a different approach, which partitions the population into "normal" types that satisfy the random sampling assumption, and "totally underdispersed" types, which are assumed to concentrate all occurrences in the corpus into a

single "burst". Using a standard LNRE model for the normal part of the population and a simple linear growth model for the underdispersed part, adjusted values for $E[V]$ and $E[V_m]$ can easily be calculated. These so-called *partition-adjusted* models (which introduce one additional parameter) are thus the only viable models for non-randomness correction in LNRE modeling and have to be considered the state of the art.

4.2 *Echo* adjustment

Rather than making more complex assumptions about the population distribution or the sampling model, we propose that non-randomness should be tackled as a *pre-processing* problem. The issue, we argue, is really with the way we count occurrences of types. The fact that a rare topic-specific word occurs, say, four times in a single document does not make it any less a hapax legomenon for our purposes than if the word occurred once (this is the case, for example, of the word *chondritic* in the BNC, which occurs 4 times, all in the same scientific document).

We operationalize our intuition by proposing that, for our purposes, each content word (at least each rare, topic-specific content word) occurs maximally once in a document, and all other instances of that word in the document are really instances of a special "anaphoric" type, whose function is that of "echoing" the content words in the document. Thus, in the BNC document mentioned above, the word *chondritic* is counted only once, whereas the other three occurrences are considered as tokens of the *echo* type. Thus, we are counting what in the information retrieval literature is known as *document frequencies*. Intuitively, these are less susceptible to topical clumpiness effects than plain token frequencies. However, by replacing repeated words with echo tokens, we can stick to a sampling model based on random word token sampling (rather than document sampling), so that the LNRE models can be applied "as is" to echo-adjusted corpora.

Echo-adjustment does not affect the sample size N nor the vocabulary size V, making the interpretation of results obtained with echo-adjusted models entirely straightforward. N does not change because repeated types are replaced with echo tokens, not deleted. V does not change because only repeated types are replaced. Thus, no type present in the original corpus disappears (more precisely, V increases by 1 because of the addition of the echo type, but given the large size of V this can be ignored for all practical purposes). Thus, the expected V computed for a specified sample size N with a model trained on an echo-adjusted corpus can be directly compared to observed values at N, and to predictions made for the same N by models trained on an unprocessed corpus. The same is not true for the prediction of the frequency distribution, where, for the same N, echo-based models predict the distribution of *document* frequencies.

We are proposing echoes as a model for the usage of (rare) content words. It would be difficult to decide where the boundary is between topical words that are inserted once in a discourse and then anaphorically modulated and "general-purpose" words that constitute the frame of the discourse and can occur multiple times. Luckily, we do not have to make this decision when estimating a LNRE model, since model fitting is based on the distribution of the lowest frequencies. For example, with the default zipfR model fitting setting, only the lowest 15 spectrum elements are used to fit the models. For any reasonably sized corpus, it is unlikely that function words and common content words will occur in less than 16 documents, and thus their distribution will be irrelevant for model fitting. Thus, we can ignore the issue of what is the boundary between topical words to be echo-adjusted and general words, as long as we can be confident that the set of lowest frequency words used for model fitting belong to the topical set.[9] This makes practical echo-adjustment extremely simple, since all we have to do is to replace all repetitions of a word in the same document with echo tokens, and estimate the parameters of a plain LNRE model with the resulting version of the training corpus.

4.3 Experiments with echo adjustment

Using the same training and test sets as in Section 3.1, we train the partition-adjusted GIGP model

[9] The issue becomes more delicate if we want to predict the frequency spectrum rather than V, since a model trained on echo-adjusted data will predict echo-adjusted frequencies across the board. However, in many theoretical and practical settings only the lowest frequency spectrum elements are of interest, where, again, it is safe to assume that words are highly topic-dependent, and echo-adjustment is appropriate.

implemented in the LEXSTATS toolkit (Baayen, 2001). We estimate the parameters of echo-adjusted ZM, fZM and GIGP models on versions of the training corpora that have been pre-processed as described above. The performance of the models is evaluated with the same measures as in Section 3.1 (for prediction of V_1, echo-adjusted versions of the test data are used).

Figure 1 reports the performance of the echo-adjusted fZM and GIGP models and of partition-adjusted GIGP (echo-adjusted ZM performed systematically much worse than the other echo-adjusted models and typically worse than uncorrected ZM, and it is not reported in the figure). Both correction methods lead to a dramatic improvement, bringing the prediction performance of fZM and GIGP to levels comparable to ZM (with the latter outperforming the corrected models on the BNC, but being outperformed on la Repubblica). Moreover, echo-adjusted GIGP is as good as partitioned GIGP on la Repubblica, and better on both BNC and deWaC, suggesting that the much simpler echo-adjustment method is at least as good and probably better than Baayen's partitioning. The mean error and confidence interval plots in Figure 2 show that the echo-adjusted models have a much weaker underestimation bias than the corresponding unadjusted models, and are comparable to, if not better than, ZM (although they might have a tendency to display more variance, as clearly illustrated by the performance of echo-adjusted fZM on la Repubblica at $3N_0$ prediction size). Finally, the echo-adjusted models clearly stand out with respect to ZM when it comes to V_1 prediction (Figure 3), indicating that echo-adjusted versions of the more sophisticated fZM and GIGP models should be the focus of future work on improving prediction of the full frequency distribution, rather than plain ZM. Moreover, echo-adjusted GIGP is outperforming partitioned GIGP, and emerging as the best model overall.[10] Reassuringly, for the echoed models there is a very strong *positive* correlation between goodness-of-fit on the training set and quality of prediction, as illustrated for V prediction at $3N_0$ by the triangles in Figure 4 (again, the patterns in this figure represent the general trend for echo-adjusted models found in all settings).[11] This indicates that the over-fitting problem has been resolved, and for echo-adjusted models goodness-of-fit on the training set is a reliable indicator of prediction accuracy.

[10] In looking at the V_1 data, it must be kept in mind, however, that V_1 has a different interpretation when predicted by echo-adjusted models, i.e., it is the number of *document-based* hapaxes, the number of types that occur in one document only.

5 Conclusion

Despite the encouraging results we reported, much work, of course, remains to be done. Even with the echo-adjusted models, prediction of V_1 suffers from large errors and prediction of V quickly deteriorates with increasing prediction size N. If the models' estimates for 3 times the size of the training set have acceptable errors of around 5%, for many applications we might want to extrapolate to $100N_0$ or more (recall the example of estimating type counts for the entire Web). Moreover, echo-adjusted models make predictions pertaining to the distribution of document frequencies, rather than plain token frequencies. The full implications of this remain to be investigated. Finally, future work should systematically explore to what extent different textual typologies are affected by the non-randomness problem (notice, e.g., that non-randomness seems to be a greater problem for the BNC than for the more uniform la Repubblica corpus).

References

Baayen, Harald. 1992. Quantitative aspects of morphological productivity. *Yearbook of Morphology 1991*, 109-150.

Baayen, Harald. 2001. *Word frequency distributions*. Dordrecht: Kluwer.

Church, Kenneth W. and William A. Gale. 1995. Poisson mixtures. *Journal of Natural Language Engineering* **1**, 163-190.

Evert, Stefan. 2004. A simple LNRE model for random character sequences. *Proceedings of JADT 2004*, 411-422.

Evert, Stefan and Marco Baroni. 2006. Testing the extrapolation quality of word frequency models. *Proceedings of Corpus Linguistics 2005*.

Katz, Slava M. 1996. Distribution of content words and phrases in text and language modeling. *Natural Language Engineering*, **2**(2) 15-59.

[11] With significant correlation coefficients of $r = .76$ for $2N_0$ ($p < 0.05$) and $r = .94$ for $3N_0$ ($p \ll 0.01$).

A System for Large-Scale Acquisition of Verbal, Nominal and Adjectival Subcategorization Frames from Corpora

Judita Preiss, Ted Briscoe, and Anna Korhonen
Computer Laboratory
University of Cambridge
15 JJ Thomson Avenue
Cambridge CB3 0FD, UK
Judita.Preiss, Ted.Briscoe, Anna.Korhonen@cl.cam.ac.uk

Abstract

This paper describes the first system for large-scale acquisition of subcategorization frames (SCFs) from English corpus data which can be used to acquire comprehensive lexicons for verbs, nouns and adjectives. The system incorporates an extensive rule-based classifier which identifies 168 verbal, 37 adjectival and 31 nominal frames from grammatical relations (GRs) output by a robust parser. The system achieves state-of-the-art performance on all three sets.

1 Introduction

Research into automatic acquisition of lexical information from large repositories of unannotated text (such as the web, corpora of published text, etc.) is starting to produce large scale lexical resources which include frequency and usage information tuned to genres and sublanguages. Such resources are critical for natural language processing (NLP), both for enhancing the performance of state-of-art statistical systems and for improving the portability of these systems between domains.

One type of lexical information with particular importance for NLP is subcategorization. Access to an accurate and comprehensive subcategorization lexicon is vital for the development of successful parsing technology (e.g. (Carroll et al., 1998), important for many NLP tasks (e.g. automatic verb classification (Schulte im Walde and Brew, 2002)) and useful for any application which can benefit from information about predicate-argument structure (e.g. Information Extraction (IE) ((Surdeanu et al., 2003)).

The first systems capable of automatically learning a small number of verbal subcategorization frames (SCFs) from unannotated English corpora emerged over a decade ago (Brent, 1991; Manning, 1993). Subsequent research has yielded systems for English (Carroll and Rooth, 1998; Briscoe and Carroll, 1997; Korhonen, 2002) capable of detecting comprehensive sets of SCFs with promising accuracy and demonstrated success in application tasks (e.g. (Carroll et al., 1998; Korhonen et al., 2003)). Recently, a large publicly available subcategorization lexicon was produced using such technology which contains frame and frequency information for over 6,300 English verbs – the VALEX lexicon (Korhonen et al., 2006).

While there has been considerable work in the area, most of it has focussed on verbs. Although verbs are the richest words in terms of subcategorization and although verb SCF distribution data is likely to offer the greatest boost in parser performance, accurate and comprehensive knowledge of the many noun and adjective SCFs in English could improve the accuracy of parsing at several levels (from tagging to syntactic and semantic analysis).

Furthermore the selection of the correct analysis from the set returned by a parser which does not initially utilize fine-grained lexico-syntactic information can depend on the *interaction* of conditional probabilities of lemmas of different classes occur-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 912–919,
Prague, Czech Republic, June 2007.

ring with specific SCFs. For example, a) and b) below indicate the most plausible analyses in which the sentential complement attaches to the noun and verb respectively

a) Kim (VP believes (NP the evidence (Scomp that Sandy was present)))

b) Kim (VP persuaded (NP the judge) (Scomp that Sandy was present))

However, both a) and b) consist of an identical sequence of coarse-grained lexical syntactic categories, so correctly ranking them requires learning that $P(NP \mid believe).P(Scomp \mid evidence) > P(NP\&Scomp \mid believe).P(None \mid evidence)$ and $P(NP \mid persuade).P(Scomp \mid judge) < P(NP\&Scomp \mid persuade).P(None \mid judge)$. If we acquired frames and frame frequencies for all open-class predicates taking SCFs using a single system applied to similar data, we would have a better chance of modeling such interactions accurately.

In this paper we present the first system for large-scale acquisition of SCFs from English corpus data which can be used to acquire comprehensive lexicons for verbs, nouns and adjectives. The classifier incorporates 168 verbal, 37 adjectival and 31 nominal SCF distinctions. An improved acquisition technique is used which expands on the ideas Yallop et al. (2005) recently explored for a small experiment on adjectival SCF acquisition. It involves identifying SCFs on the basis of grammatical relations (GRs) in the output of the RASP (Robust Accurate Statistical Parsing) system (Briscoe et al., 2006).

As detailed later, the system performs better with verbs than previous comparable state-of-art systems, achieving 68.9 F-measure in detecting SCF types. It achieves similarly good performance with nouns and adjectives (62.2 and 71.9 F-measure, respectively).

Additionally, we have developed a tool for linguistic annotation of SCFs in corpus data aimed at alleviating the process of obtaining training and test data for subcategorization acquisition. The tool incorporates an intuitive interface with the ability to significantly reduce the number of frames presented to the user for each sentence.

We introduce the new system for SCF acquisition in section 2. Details of the experimental evaluation are supplied in section 3. Section 4 provides discussion of our results and future work, and section 5 concludes.

2 Description of the System

A common strategy in existing large-scale SCF acquisition systems (e.g. (Briscoe and Carroll, 1997)) is to extract SCFs from parse trees, introducing an unnecessary dependence on the details of a particular parser. In our approach SCFs are extracted from GRs — representations of head-dependent relations which are more parser/grammar independent but at the appropriate level of abstraction for extraction of SCFs.

A similar approach was recently motivated and explored by Yallop et al. (2005). A decision-tree classifier was developed for 30 adjectival SCF types which tests for the presence of GRs in the GR output of the RASP (Robust Accurate Statistical Parsing) system (Briscoe and Carroll, 2002). The results reported with 9 test adjectives were promising (68.9 F-measure in detecting SCF types).

Our acquisition process consists of four main steps: 1) extracting GRs from corpus data, 2) feeding the GR sets as input to a rule-based classifier which incrementally matches them with the corresponding SCFs, 3) building lexical entries from the classified data, and 4) filtering those entries to obtain a more accurate lexicon. The details of these steps are provided in the subsequent sections.

2.1 Obtaining Grammatical Relations

We obtain the GRs using the recent, second release of the RASP toolkit (Briscoe et al., 2006). RASP is a modular statistical parsing system which includes a tokenizer, tagger, lemmatizer, and a wide-coverage unification-based tag-sequence parser. We use the standard scripts supplied with RASP to output the set of GRs for the most probable analysis returned by the parser or, in the case of parse failures, the GRs for the most likely sequence of subanalyses. The GRs are organized as a subsumption hierarchy as shown in Figure 1.

The dependency relationships which the GRs embody correspond closely to the head-complement structure which subcategorization acquisition attempts to recover, which makes GRs ideal input to the SCF classifier. Consider the arguments of *easy*

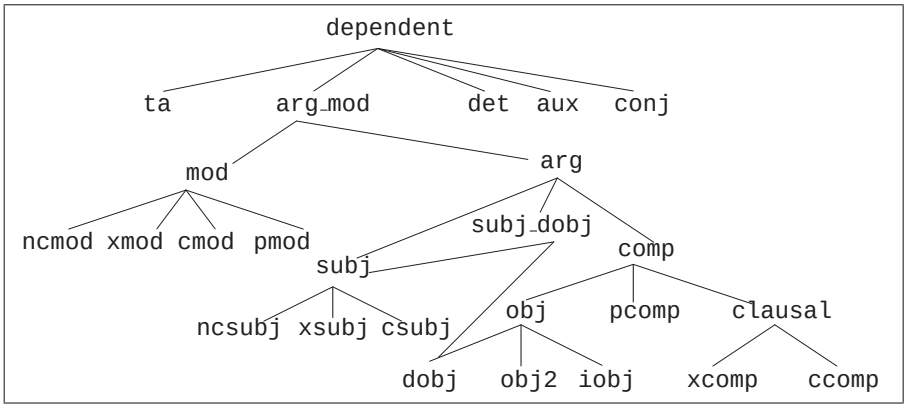


Figure 1: The GR hierarchy used by RASP

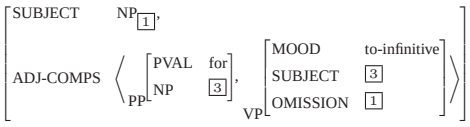


Figure 2: Feature structure for SCF `adj-obj-for-to-inf`

```
(|These:1_DD2| |example+s:2_NN2| |of:3_IO|
 |animal:4_JJ| |senses:5_NN2| |be+:6_VBR|
 |relatively:7_RR| |easy:8_JJ| |for:9_IF|
 |we+:10_PPIO2| |to:11_TO| |comprehend:12_VV0|)
...
 xcomp(_               be+[6]       easy:[8])
 xcomp(to[11]          be+[6]       comprehend:[12])
ncsubj(be+[6]          example+s[2] _)
 ncmod(for[9]          easy[8]      we+[10])
ncsubj(comprehend[12] we+[10], _)
...
```

Figure 3: GRs from RASP for `adj-obj-for-to-inf`

in the sentence: *These examples of animal senses are relatively easy for us to comprehend as they are not too far removed from our own experience.* According to the COMLEX classification, this is an example of the frame `adj-obj-for-to-inf`, shown in Figure 2, (using AVM notation in place of COMLEX s-expressions). Part of the output of RASP for this sentence is shown in Figure 3.

Each instantiated GR in Figure 3 corresponds to one or more parts of the feature structure in Figure 2. `xcomp(_ be[6] easy[8])` establishes `be[6]` as the head of the `VP` in which `easy[8]` occurs as a complement. The first (`PP`)-complement is *for us*, as indicated by `ncmod(for[9] easy[8] we+[10])`, with *for* as `PFORM` and `we+` (*us*) as `NP`. The second complement is represented by `xcomp(to[11] be+[6] comprehend[12])`: a *to-infinitive* `VP`. The

xcomp	_	?Y : pos=vb,val=be	?X : pos=adj
xcomp	?S : val=to	?Y : pos=vb,val=be	?W : pos=VV0
ncsubj	?Y : pos=vb,val=be	?Z : pos=noun	
ncmod	?T : val=for	?X : pos=adj	?Y: pos=pron
ncsubj	?W : pos=VV0	?V : pos=pron	

Figure 4: Pattern for frame `adj-obj-for-to-inf`

`NP` headed by *examples* is marked as the subject of the frame by `ncsubj(be[6] examples[2])`, and `ncsubj(comprehend[12] we+[10])` corresponds to the coindexation marked by 3: the subject of the `VP` is the `NP` of the `PP`. The only part of the feature structure which is not represented by the GRs is coindexation between the omitted direct object 1 of the `VP`-complement and the subject of the whole clause.

2.2 SCF Classifier

SCF Frames

The SCFs recognized by the classifier were obtained by manually merging the frames exemplified in the COMLEX Syntax (Grishman et al., 1994), ANLT (Boguraev et al., 1987) and/or NOMLEX (Macleod et al., 1997) dictionaries and including additional frames found by manual inspection of unclassifiable examples during development of the classifier. These consisted of e.g. some occurrences of phrasal verbs with complex complementation and with flexible ordering of the preposition/particle, some non-passivizable words with a surface direct object, and some rarer combinations of governed preposition and complementizer combinations.

The frames were created so that they abstract over specific lexically-governed particles and prepositions and specific predicate selectional preferences

but include some derived semi-predictable bounded dependency constructions.

Classifier

The classifier operates by attempting to match the set of GRs associated with each sentence against one or more rules which express the possible mappings from GRs to SCFs. The rules were manually developed by examining a set of development sentences to determine which relations were actually emitted by the parser for each SCF.

In our rule representation, a GR pattern is a set of partially instantiated GRs with variables in place of heads and dependents, augmented with constraints that restrict the possible instantiations of the variables. A match is successful if the set of GRs for a sentence can be unified with any rule. Unification of sentence GRs and a rule GR pattern occurs when there is a one-to-one correspondence between sentence elements and rule elements that includes a consistent mapping from variables to values.

A sample pattern for matching `adj-obj-for-to-inf` can be seen in Figure 4. Each element matches either an empty GR slot (_), a variable with possible constraints on part of speech (pos) and word value (val), or an already instantiated variable. Unlike in Yallop's work (Yallop et al., 2005), our rules are declarative rather than procedural and these rules, written independently of the acquisition system, are expanded by the system in a number of ways prior to execution. For example, the verb rules which contain an `ncsubj` relation will not contain one inside an embedded clause. For verbs, the basic rule set contains 248 rules but automatic expansion gives rise to 1088 classifier rules for verbs.

Numerous approaches were investigated to allow an efficient execution of the system: for example, for each target word in a sentence, we initially find the number of ARGument GRs (see Figure 1) containing it in head position, as the word must appear in exactly the same set in a matching rule. This allows us to discard all patterns which specify a different number of GRs: for example, for verbs each group only contains an average of 109 patterns.

For a further increase in speed, both the sentence GRs and the GRs within the patterns are ordered (according to frequency) and matching is performed using a backing off strategy allowing us to exploit the relatively low number of possible GRs (compared to the number of possible rules). The system executes on 3500 sentences in approx. 1.5 seconds of real time on a machine with a 3.2 GHz Intel Xenon processor and 4GB of RAM.

Lexicon Creation and Filtering

Lexical entries are constructed for each word and SCF combination found in the corpus data. Each lexical entry includes the raw and relative frequency of the SCF with the word in question, and includes various additional information e.g. about the syntax of detected arguments and the argument heads in different argument positions[1].

Finally the entries are filtered to obtain a more accurate lexicon. A way to maximise the accuracy of the lexicon would be to smooth (correct) the acquired SCF distributions with back-off estimates based on lexical-semantic classes of verbs (Korhonen, 2002) (see section 4) before filtering them. However, in this first experiment with the new system we filtered the entries directly so that we could evaluate the performance of the new classifier without any additional modules. For the same reason, the filtering was done by using a very simple method: by setting empirically determined thresholds on the relative frequencies of SCFs.

3 Experimental Evaluation

3.1 Data

In order to test the accuracy of our system, we selected a set of 183 verbs, 30 nouns and 30 adjectives for experimentation. The words were selected at random, subject to the constraint that they exhibited multiple complementation patterns and had a sufficient number of corpus occurrences (> 150) for experimentation. We took the 100M-word British National Corpus (BNC) (Burnard, 1995), and extracted all sentences containing an occurrence of one of the test words. The sentences were processed using the SCF acquisition system described in the previous section. The citations from which entries were derived totaled approximately 744K for verbs and 219K for nouns and adjectives, respectively.

[1]The lexical entries are similar to those in the VALEX lexicon. See (Korhonen et al., 2006) for a sample entry.

3.2 Gold Standard

Our gold standard was based on a manual analysis of some of the test corpus data, supplemented with additional frames from the ANLT, COMLEX, and/or NOMLEX dictionaries. The gold standard for verbs was available, but it was extended to include additional SCFs missing from the old system. For nouns and adjectives the gold standard was created. For each noun and adjective, 100-300 sentences from the BNC (an average of 267 per word) were randomly extracted. The resulting c. 16K sentences were then manually associated with appropriate SCFs, and the SCF frequency counts were recorded.

To alleviate the manual analysis we developed a tool which first uses the RASP parser with some heuristics to reduce the number of SCF presented, and then allows an annotator to select the preferred choice in a window. The heuristics reduced the average number of SCFs presented alongside each sentence from 52 to 7. The annotator was also presented with an example sentence of each SCF and an intuitive name for the frame, such as PRED (e.g. *Kim is silly*). The program includes an option to record that particular sentences could not (initially) be classified. A screenshot of the tool is shown in Figure 5.

The manual analysis was done by two linguists; one who did the first annotation for the whole data, and another who re-evaluated and corrected some of the initial frame assignments, and classified most of the data left unclassified by the first annotator[2]). A total of 27 SCF types were found for the nouns and 30 for the adjectives in the annotated data. The average number of SCFs taken by nouns was 9 (with the average of 2 added from dictionaries to supplement the manual annotation) and by adjectives 11 (3 of which were from dictionaries). The latter are rare and may not be exemplified in the data given the extraction system.

3.3 Evaluation Measures

We used the standard evaluation metrics to evaluate the accuracy of the SCF lexicons: type precision (the percentage of SCF types that the system proposes which are correct), type recall (the percentage of SCF types in the gold standard that the system proposes) and the F-measure which is the harmonic mean of type precision and recall.

[2]The process precluded measurements of inter-annotator agreement, but this was judged less important than the enhanced accuracy of the gold standard data.

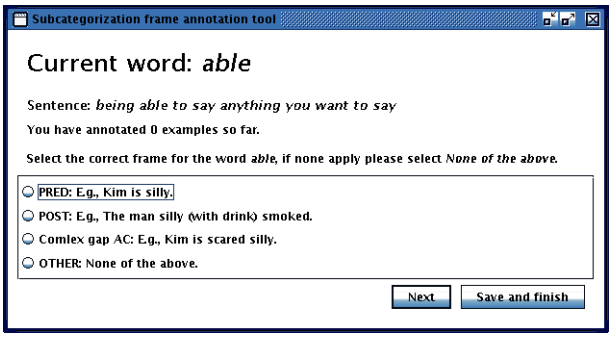


Figure 5: Sample screen of the annotation tool

We also compared the similarity between the acquired unfiltered[3] SCF distributions and gold standard SCF distributions using various measures of distributional similarity: the Spearman rank correlation (RC), Kullback-Leibler distance (KL), Jensen-Shannon divergence (JS), cross entropy (CE), skew divergence (SD) and intersection (IS). The details of these measures and their application to subcategorization acquisition can be found in (Korhonen and Krymolowski, 2002).

Finally, we recorded the total number of gold standard SCFs unseen in the system output, i.e. the type of false negatives which were never detected by the classifier.

3.4 Results

Table 1 includes the average results for the 183 verbs. The first column shows the results for Briscoe and Carroll's (1997) (B&C) system when this system is run with the original classifier but a more recent version of the parser (Briscoe and Carroll, 2002) and the same filtering technique as our new system (thresholding based on the relative frequencies of SCFs). The classifier of B&C system is comparable to our classifier in the sense that it targets almost the same set of verbal SCFs (165 out of the 168; the 3 additional ones are infrequent in language and thus unlikely to affect the comparison). The second column shows the results for our new system (New).

[3]No threshold was applied to remove the noisy SCFs from the distributions.

Measures	Verbs - Method	
	B&C	New
Precision (%)	47.3	81.8
Recall (%)	40.4	59.5
F-measure	43.6	68.9
KL	3.24	1.57
JS	0.20	0.11
CE	4.85	3.10
SD	1.39	0.74
RC	0.33	0.66
IS	0.49	0.76
Unseen SCFs	28	17

Table 1: Average results for verbs

Measures	Nouns	Adjectives
Precision (%)	91.2	95.5
Recall (%)	47.2	57.6
F-measure	62.2	71.9
KL	0.91	0.69
JS	0.09	0.05
CE	2.03	2.01
SD	0.48	0.36
RC	0.70	0.77
IS	0.62	0.72
Unseen SCFs	15	7

Table 2: Average results for nouns and adjectives

The figures show that the new system clearly performs better than the B&C system. It yields 68.9 F-measure which is a 25.3 absolute improvement over the B&C system. The better performance can be observed on all measures, but particularly on SCF type precision (81.8% with our system vs. 47.3% with the B&C system) and on measures of distributional similarity. The clearly higher IS (0.76 vs. 0.49) and the fewer gold standard SCFs unseen in the output of the classifier (17 vs. 28) indicate that the new system is capable of detecting a higher number of SCFs.

The main reason for better performance is the ability of the new system to detect a number of challenging or complex SCFs which the B&C system could not detect[4]. The improvement is partly attributable to more accurate parses produced by the second release of RASP and partly to the improved SCF classifier developed here. For example, the new system is now able to distinguish predicative PP arguments, such as *I sent him as a messenger* from the wider class of referential PP arguments, supporting discrimination of several syntactically similar SCFs with distinct semantics.

Running our system on the adjective and noun test data yielded the results summarized in Table 2. The F-measure is lower for nouns (62.2) than for verbs (68.9); for adjectives it is slightly better (71.9).[5]

The noun and adjective classifiers yield very high precision compared to recall. The lower recall figures are mostly due to the higher number of gold standard SCFs unseen in the classifier output (rather than, for example, the filtering step). This is particularly evident for nouns for which 15 of the 27 frames exemplified in the gold standard are missing in the classifier output. For adjectives only 7 of the 30 gold standard SCFs are unseen, resulting in better recall (57.6% vs. 47.2% for nouns).

For verbs, subcategorization acquisition performance often correlates with the size of the input data to acquisition (the more data, the better performance). When considering the F-measure results for the individual words shown in Table 3 there appears to be little such correlation for nouns and adjectives. For example, although there are individual high frequency nouns with high performance (e.g. *plan*, freq. 5046, F 90.9) and low frequency nouns with low performance (e.g. *characterisation*, freq. 91, F 40.0), there are also many nouns which contradict the trend (compare e.g. *answer*, freq. 2510, F 50.0 with *fondness*, freq. 71, F 85.7).[6]

Although the SCF distributions for nouns and adjectives appear Zipfian (i.e. the most frequent frames are highly probable, but most frames are infrequent), the total number of SCFs per word is typically smaller than for verbs, resulting in better resistance to sparse data problems.

There is, however, a clear correlation between the performance and the type of gold standard SCFs taken by individual words. Many of the gold stan-

[4]The results reported here for the B&C system are lower than those recently reported in (Korhonen et al., 2006) for the same set of 183 test verbs. This is because we use an improved gold standard. However, the results for the B&C system reported using the less ambitious gold standard are still less accurate (58.6 F-measure) than the ones reported here for the new system.

[5]The results for different word classes are not directly comparable because they are affected by the total number of SCFs evaluated for each word class, which is higher for verbs and lower for nouns and adjectives. This particularly applies to the sensitive measures of distributional similarity.

[6]The frequencies here refer to the number of citations successfully processed by the parser and the classifier.

Noun	F	Adjective	F
abundance	75.0	*able*	66.7
acknowledgement	47.1	*angry*	62.5
answer	50.0	*anxious*	82.4
anxiety	53.3	*aware*	87.5
apology	50.0	*certain*	73.7
appearance	46.2	*clear*	77.8
appointment	66.7	*curious*	57.1
belief	76.9	*desperate*	83.3
call	58.8	*difficult*	77.8
characterisation	40.0	*doubtful*	63.6
communication	40.0	*eager*	83.3
condition	66.7	*easy*	66.7
danger	76.9	*generous*	57.1
decision	70.6	*imperative*	81.8
definition	42.8	*important*	60.9
demand	66.7	*impractical*	71.4
desire	71.4	*improbable*	54.6
doubt	66.7	*insistent*	80.0
evidence	66.7	*kind*	66.7
examination	54.6	*likely*	66.7
experimentation	60.0	*practical*	88.9
fondness	85.7	*probable*	80.0
message	66.7	*sure*	84.2
obsession	54.6	*unaware*	85.7
plan	90.9	*uncertain*	60.0
provision	70.6	*unclear*	63.2
reminder	63.2	*unimportant*	61.5
rumour	61.5	*unlikely*	69.6
temptation	71.4	*unspecified*	50.0
use	60.0	*unsure*	90.0

Table 3: System performance for each test noun and adjective

dard nominal and adjectival SCFs unseen by the classifier involve complex complementation patterns which are challenging to extract, e.g. those exemplified in *The* **argument** *of Jo with Kim about Fido surfaced, Jo's* **preference** *that Kim be sacked surfaced*, and *that Sandy came is* **certain**. In addition, many of these SCFs unseen in the data are also very low in frequency, and some may even be true negatives (recall that the gold standard was supplemented with additional SCFs from dictionaries, which may not necessarily appear in the test data).

The main problem is that the RASP parser systematically fails to select the correct analysis for some SCFs with nouns and adjectives regardless of their context of occurrence. In future work, we hope to alleviate this problem by using the weighted GR output from the top *n*-ranked parses returned by the parser as input to the SCF classifier.

4 Discussion

The current system needs refinement to alleviate the bias against some SCFs introduced by the parser's unlexicalized parse selection model. We plan to investigate using weighted GR output with the classifier rather than just the GR set from the highest ranked parse. Some SCF classes also need to be further resolved mainly to differentiate control options with predicative complementation. This requires a lexico-semantic classification of predicate classes.

Experiments with Briscoe and Carroll's system have shown that it is possible to incorporate some semantic information in the acquisition process using a technique that smooths the acquired SCF distributions using back-off (i.e. probability) estimates based on lexical-semantic classes of verbs (Korhonen, 2002). The estimates help to correct the acquired SCF distributions and predict SCFs which are rare or unseen e.g. due to sparse data. They could also form the basis for predicting control of predicative complements.

We plan to modify and extend this technique for the new system and use it to improve the performance further. The technique has so far been applied to verbs only, but it can also be applied to nouns and adjectives because they can also be classified on lexical-semantic grounds. For example, the adjective *simple* belongs to the class of EASY adjectives, and this knowledge can help to predict that it takes similar SCFs to the other class members and that control of 'understood' arguments will pattern with *easy* (e.g. *easy, difficult, convenient*): *The problem will be simple for John to solve*, *For John to solve the problem will be simple*, *The problem will be simple to solve*, etc.

Further research is needed before highly accurate lexicons encoding information also about semantic aspects of subcategorization (e.g. different predicate senses, the mapping from syntactic arguments to semantic representation of argument structure, selectional preferences on argument heads, diathesis alternations, etc.) can be obtained automatically. However, with the extensions suggested above, the system presented here is sufficiently accurate for building an extensive SCF lexicon capable of supporting various NLP application tasks. Such a lexicon will be built and distributed for research pur-

poses along with the gold standard described here.

5 Conclusion

We have described the first system for automatically acquiring verbal, nominal and adjectival subcategorization and associated frequency information from English corpora, which can be used to build large-scale lexicons for NLP purposes. We have also described a new annotation tool for producing training and test data for the task. The acquisition system, which is capable of distinguishing 168 verbal, 37 adjectival and 31 nominal frames, classifies corpus occurrences to SCFs on the basis of GRs produced by a robust statistical parser. The information provided by GRs closely matches the structure that subcategorization acquisition seeks to recover. Our experiment shows that the system achieves state-of-the-art performance with each word class. The discussion suggests ways in which we could improve the system further before using it to build a large subcategorization lexicon capable of supporting various NLP application tasks.

Acknowledgements

This work was supported by the Royal Society and UK EPSRC project 'Accurate and Comprehensive Lexical Classification for Natural Language Processing Applications' (ACLEX). We would like to thank Diane Nicholls for her help during this work.

References

B. Boguraev, J. Carroll, E. J. Briscoe, D. Carter, and C. Grover. 1987. The derivation of a grammatically-indexed lexicon from the Longman Dictionary of Contemporary English. In *Proc. of the 25th Annual Meeting of ACL*, pages 193–200, Stanford, CA.

M. Brent. 1991. Automatic acquisition of subcategorization frames from untagged text. In *Proc. of the 29th Meeting of ACL*, pages 209–214.

E. J. Briscoe and J. Carroll. 1997. Automatic Extraction of Subcategorization from Corpora. In *Proc. of the 5th ANLP*, Washington DC, USA.

E. J. Briscoe and J. Carroll. 2002. Robust accurate statistical annotation of general text. In *Proc. of the 3rd LREC*, pages 1499–1504, Las Palmas, Canary Islands, May.

E. J. Briscoe, J. Carroll, and R. Watson. 2006. The second release of the rasp system. In *Proc. of the COLING/ACL 2006 Interactive Presentation Sessions*, Sydney, Australia.

L. Burnard, 1995. *The BNC Users Reference Guide*. British National Corpus Consortium, Oxford, May.

G. Carroll and M. Rooth. 1998. Valence induction with a head-lexicalized pcfg. In *Proc. of the 3rd Conference on EMNLP*, Granada, Spain.

J. Carroll, G. Minnen, and E. J. Briscoe. 1998. Can Subcategorisation Probabilities Help a Statistical Parser? In *Proceedings of the 6th ACL/SIGDAT Workshop on Very Large Corpora*, pages 118–126, Montreal, Canada.

R. Grishman, C. Macleod, and A. Meyers. 1994. COMLEX Syntax: Building a Computational Lexicon. In *COLING*, Kyoto.

A. Korhonen and Y. Krymolowski. 2002. On the Robustness of Entropy-Based Similarity Measures in Evaluation of Subcategorization Acquisition Systems. In *Proc. of the Sixth CoNLL*, pages 91–97, Taipei, Taiwan.

A. Korhonen, Y. Krymolowski, and Z. Marx. 2003. Clustering Polysemic Subcategorization Frame Distributions Semantically. In *Proc. of the 41st Annual Meeting of ACL*, pages 64–71, Sapporo, Japan.

A. Korhonen, Y. Krymolowski, and E. J. Briscoe. 2006. A large subcategorization lexicon for natural language processing applications. In *Proc. of the 5th LREC*, Genova, Italy.

A. Korhonen. 2002. *Subcategorization acquisition*. Ph.D. thesis, University of Cambridge Computer Laboratory.

C. Macleod, A. Meyers, R. Grishman, L. Barrett, and R. Reeves. 1997. Designing a dictionary of derived nominals. In *Proc. of RANLP*, Tzigov Chark, Bulgaria.

C. Manning. 1993. Automatic Acquisition of a Large Subcategorization Dictionary from Corpora. In *Proc. of the 31st Meeting of ACL*, pages 235–242.

S. Schulte im Walde and C. Brew. 2002. Inducing german semantic verb classes from purely syntactic subcategorisation information. In *Proc. of the 40th Annual Meeting of ACL*, Philadephia, USA.

M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proc. of the 41st Annual Meeting of ACL*, Sapporo.

J. Yallop, A. Korhonen, and E. J. Briscoe. 2005. Automatic acquisition of adjectival subcategorization from corpora. In *Proc. of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 614–621, Ann Arbor, Michigan.

A Language-Independent Unsupervised Model for Morphological Segmentation

Vera Demberg
School of Informatics
University of Edinburgh
Edinburgh, EH8 9LW, GB
v.demberg@sms.ed.ac.uk

Abstract

Morphological segmentation has been shown to be beneficial to a range of NLP tasks such as machine translation, speech recognition, speech synthesis and information retrieval. Recently, a number of approaches to unsupervised morphological segmentation have been proposed. This paper describes an algorithm that draws from previous approaches and combines them into a simple model for morphological segmentation that outperforms other approaches on English and German, and also yields good results on agglutinative languages such as Finnish and Turkish. We also propose a method for detecting variation within stems in an unsupervised fashion. The segmentation quality reached with the new algorithm is good enough to improve grapheme-to-phoneme conversion.

1 Introduction

Morphological segmentation has been shown to be beneficial to a number of NLP tasks such as machine translation (Goldwater and McClosky, 2005), speech recognition (Kurimo et al., 2006), information retrieval (Monz and de Rijke, 2002) and question answering. Segmenting a word into meaning-bearing units is particularly interesting for morphologically complex languages where words can be composed of several morphemes through inflection, derivation and composition. Data sparseness for such languages can be significantly decreased when words are decomposed morphologically. There exist a number of rule-based morphological segmentation systems for a range of languages. However, expert knowledge and labour are expensive, and the analyzers must be updated on a regular basis in order to cope with language change (the emergence of new words and their inflections). One might argue that unsupervised algorithms are not an interesting option from the engineering point of view, because rule-based systems usually lead to better results. However, segmentations from an unsupervised algorithm that is language-independent are "cheap", because the only resource needed is unannotated text. If such an unsupervised system reaches a performance level that is good enough to help another task, it can constitute an attractive additional component.

Recently, a number of approaches to unsupervised morphological segmentation have been proposed. These algorithms autonomously discover morpheme segmentations in unannotated text corpora. Here we describe a modification of one such unsupervised algorithm, RePortS (Keshava and Pitler, 2006). The RePortS algorithm performed best on English in a recent competition on unsupervised morphological segmentation (Kurimo et al., 2006), but had very low recall on morphologically more complex languages like German, Finnish or Turkish. We add a new step designed to achieve higher recall on morphologically complex languages and propose a method for identifying related stems that underwent regular non-concatenative morphological processes such as umlauting or ablauting, as well as morphological alternations along morpheme boundaries.

The paper is structured as follows: Section

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 920–927,
Prague, Czech Republic, June 2007.

2 discusses the relationship between language-dependency and the level of supervision of a learning algorithm. We then give an outline of the main steps of the RePortS algorithm in section 3 and explain the modifications to the original algorithm in section 4. Section 5 compares results for different languages, quantifies the gains from the modifications on the algorithm and evaluates the algorithm on a grapheme-to-phoneme conversion task. We finally summarize our results in section 6.

2 Previous Work

The world's languages can be classified according to their morphology into isolating languages (little or no morphology, e.g. Chinese), agglutinative languages (where a word can be decomposed into a large number of morphemes, e.g. Turkish) and inflectional languages (morphemes are fused together, e.g. Latin).

Phenomena that are difficult to cope with for many of the unsupervised algorithms are non-concatenative processes such as vowel harmonization, ablauting and umlauting, or modifications at the boundaries of morphemes, as well as infixation (e.g. in Tagalog: sulat 'write', s-um-ulat 'wrote', s-in-ulat 'was written'), circumfixation (e.g. in German: mach-en 'do', ge-mach-t 'done'), the Arabic broken plural or reduplications (e.g. in Pingelapese: mejr 'to sleep', mejmejr 'sleeping', mejmejmejr 'still sleeping'). For words that are subject to one of the above processes it is not trivial to automatically group related words and detect regular transformational patterns.

A range of automated algorithms for morphological analysis cope with concatenative phenomena, and base their mechanics on statistics about hypothesized stems and affixes. These approaches can be further categorized into ones that use conditional entropy between letters to detect segment boundaries (Harris, 1955; Hafer and Weiss, 1974; Déjean, 1998; Monson et al., 2004; Bernhard, 2006; Keshava and Pitler, 2006; Bordag, 2006), approaches that use minimal description length and thereby minimize the size of the lexicon as measured in entries and links between the entries to constitute a word form (Goldsmith, 2001; Creutz and Lagus, 2006). These two types of approaches very closely tie the orthographic form of the word to the morphemes. They are thus not well-suited for coping with stem changes or modifications at the edges of morphemes. Only very few approaches have addressed word internal variations (Yarowski and Wicentowski, 2000; Neuvel and Fulop, 2002).

A popular and effective approach for detecting inflectional paradigms and filter affix lists is to cluster together affixes or regular transformational patterns that occur with the same stem (Monson et al., 2004; Goldsmith, 2001; Gaussier, 1999; Schone and Jurafsky, 2000; Yarowski and Wicentowski, 2000; Neuvel and Fulop, 2002; Jacquemin, 1997). We draw from this idea of clustering in order to detect orthographic variants of stems; see Section 4.3.

A few approaches also take into account syntactic and semantic information from the context the word occurs (Schone and Jurafsky, 2000; Bordag, 2006; Yarowski and Wicentowski, 2000; Jacquemin, 1997). Exploiting semantic and syntactic information is very attractive because it adds an additional dimension, but these approaches have to cope with more severe data sparseness issues than approaches that emphasize word-internal cues, and they can be computationally expensive, especially when they use LSA.

The original RePortS algorithm assumes morphology to be concatenative, and specializes on prefixation and suffixation, like most of the above approaches, which were developed and implemented for English (Goldsmith, 2001; Schone and Jurafsky, 2000; Neuvel and Fulop, 2002; Yarowski and Wicentowski, 2000; Gaussier, 1999). However, many languages are morphologically more complex. For example in German, an algorithm also needs to cope with compounding, and in Turkish words can be very long and complex. We therefore extended the original RePortS algorithm to be better adapted to complex morphology and suggest a method for coping with stem variation. These modifications render the algorithm more language-independent and thereby make it attractive for applying to other languages as well.

3 The RePortS Algorithm

On English, the RePortS algorithm clearly outperformed all other systems in Morpho Challenge

2005[1] (Kurimo et al., 2006), obtaining an F-measure of 76.8% (76.2% prec., 77.4% recall). The next best system obtained an F-score of 69%. However, the algorithm does not perform as well on other languages (Turkish, Finnish, German) due to low recall (see (Keshava and Pitler, 2006) and (Demberg, 2006), p. 47).

There are three main steps in the algorithm. First, the data is structured in two trees, which provide the basis for efficient calculation of transitional probabilities of a letter given its context. The second step is the affix acquisition step, during which a set of morphemes is identified from the corpus data. The third step uses these morphemes to segment words.

3.1 Data Structure

The data is stored in two trees, the forward tree and the backward tree. Branches correspond to letters, and nodes are annotated with the total corpus frequency of the letter sequence from the root of the tree up to the node. During the affix identification process, the forward tree is used for discovering suffixes by calculating the probability of seeing a certain letter given the previous letters of the word. The backward tree is used to determine the probability of a letter given the following letters of a word in order to find prefixes. If the transitional probability is high, the word should not be split, whereas low probability is a good indicator of a morpheme boundary. In such a tree, stems tend to stay together in long unary branches, while the branching factor is high in places where morpheme boundaries occur.

The underlying idea of exploiting "Letter Successor Variety" was first proposed in (Harris, 1955), and has since been used in a number of morphemic segmentation algorithms (Hafer and Weiss, 1974; Bernhard, 2006; Bordag, 2006).

3.2 Finding Affixes

The second step is concerned with finding good affixes. The procedure is quite simple and can be divided into two subtasks. (1) generating all possible affixes and (2) validating them. The validation step is necessary to exclude bad affix candidates (e.g. letter sequences that occur together frequently such as *sch*, *spr* or *ch* in German or *sh*, *th*, *qu* in English). An affix is validated if all three criteria are satisfied for at least 5% of its occurrences:

1. The substring that remains after peeling off an affix is also a word in the lexicon.
2. The transitional probability between the second-last and the last stem letter is ≈ 1.
3. The transitional probability of the affix letter next to the stem is <1 (tolerance 0.02).

Finally, all affixes that are concatenations of two or more other suffixes (e.g., *-ungen* can be split up in *-ung* and *-en* in German) are removed. This step returns two lists of morphological segments. The prefix list contains prefixes as well as stems that usually occur at the beginning of words, while the suffix list contains suffixes and stems that occur at the end of words. In the remainder of the paper, we will refer to the content of these lists as "prefixes" and "suffixes", although they also include stems. There are several assumptions encoded in this procedure that are specific to English, and cause recall to be low for other languages: 1) all stems are valid words in the lexicon; 2) affixes occur at the beginning or end of words only; and 3) affixation does not change stems. In section 4, we propose ways of relaxing these assumptions to make this step less language-specific.

3.3 Segmenting Words

The final step is the complete segmentation of words given the list of affixes acquired in the previous step. The original RePortS algorithm uses a very simple method that peels off the most probable suffix that has a transitional probability smaller than 1, until no more affixes match or until less than half of the word remains. This last condition is problematic since it does not scale up well to languages with complex morphology. The same peeling-off process is executed for prefixes.

Although this method is better than using a heuristic such as 'always peel off the longest possible affix', because it takes into account probable sites of fractures in words, it is not sensitive to the affix context or the morphotactics of the language. Typical mistakes that arise from this condition are that inflectional suffixes, which can only occur word-finally, might be split off in the middle of a word after previously having peeled off a number of other suffixes.

[1] www.cis.hut.fi/morphochallenge2005/

4 Modifications and Extensions

4.1 Morpheme Acquisition

When we ran the original algorithm on a German data set, no suffixes were validated but reasonable prefix lists were found. The algorithm works fine for English suffixes – why does it fail on German? The algorithm's failure to detect German suffixes is caused by the invalid assumption that a stem must be a word in the corpus. German verb stems do not occur on their own (except for certain imperative forms). After stripping off the suffix of the verb *abholst* 'fetch', the remaining string *abhol* cannot be found in the lexicon. However, words like *abholen*, *abholt*, *abhole* or *Abholung* are part of the corpus. The same problem also occurs for German nouns.

Therefore, this first condition of the affix acquisition step needs to be replaced. We therefore introduced an additional step for building an intermediate stem candidate list into the affix acquisition process. The first condition is replaced by a condition that checks whether a stem is in the stem candidate list. This new stem candidate acquisition procedure comprises three steps:

Step 1: Creation of stem candidate list

All substrings that satisfy conditions 2 and 3 but not condition 1, are stored together with the set of affixes they occur with. This process is similar to the idea of registering signatures (Goldsmith, 2001; Neuvel and Fulop, 2002). For example, let us assume our corpus contains the words *Aufführender*, *Aufführung*, *aufführt* and *Aufführlaune* but not the stem itself, since *aufführ* 'act' is not a valid German word. Conditions 2 and 3 are met, because the transitional probability between *aufführ* and the next letter is low (there are a lot of different possible continuations) and the transitional probability $P(r|auffüh) \approx 1$. The stem candidate *aufführ* is then stored together with the suffix candidates $\{$*ender*, *ung*, *en*, *t*, *laune*$\}$.

Step 2: Ranking candidate stems

There are two types of affix candidates: type-1 affix candidates are words that are contained in the data base as full words (those are due to compounding); type-2 affix candidates are inflectional and derivational suffixes. When ranking the stem candidates, we take into account the number of type-1 affix candidates and the average frequency of tpye-2 affix candidates.

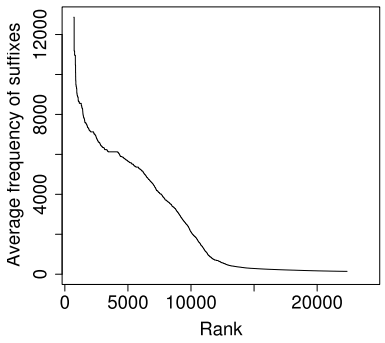


Figure 1: Determining the threshold for validating the best candidates from the stem candidate list.

The first condition has very good precision, similar to the original method. The morphemes found with this method are predominantly stem forms that occur in compounding or derivation (Kompositionsstämme and Derivationsstämme). The second condition enables us to differentiate between stems that occur with common suffixes (and therefore have high average frequencies), and pseudostems such as *runtersch* whose affix list contains many non-morphemes (e.g. *lucken*, *iebt*, *aute*). These non-morphemes are very rare since they are not generated by a regular process.

Step 3: Pruning

All stem candidates that occur less than three times are removed from the list. The remaining stem candidates are ordered according to the average frequency of their non-word suffixes. This criterion puts the high quality stem candidates (that occur with very common suffixes) to the top of the list. In order to obtain a high-precision stem list, it is necessary to cut the list of candidates at some point. The threshold for this is determined by the data: we choose the point at which the function of list-rank vs. score changes steepness (see Figure 1). This visual change of steepness corresponds to the point where potential stems found get more noisy because the strings with which they occur are not common affixes. We found the performance of the resulting morphological system to be quite stable ($\pm 1\%$ f-score) for any cutting point on the slope between 20% and 50% of the list (for the German data set ranks 4000 and 12000), but importantly before the function tails off. The threshold was also robust across the other languages and data sets.

4.2 Morphological Segmentation

As discussed in section 3.3, the original implementation of the algorithm iteratively chops off the most probable affixes at both edges of the word without taking into account the context of the affix. In morphologically complex languages, this context-blind approach often leads to suboptimal results, and also allows segmentations that are morphotactically impossible, such as inflectional suffixes in the middle of words. Another risk is that the letter sequence that is left after removing potential prefixes and suffixes from both ends is not a proper stem itself but just a single letter or vowel-less letter-sequence.

These problems can be solved by using a bi-gram language model to capture the morphotactic properties of a particular language. Instead of simply peeling off the most probable affixes from both ends of the word, all possible segmentations of the word are generated and ranked using the language model. The probabilities for the language model are learnt from a set of words that were segmented with the original simple approach. This bootstrapping allows us to ensure that the approach remains fully unsupervised. At the beginning and end of each word, an edge marker '#' is attached to the word. The model can then also acquire probabilities about which affixes occur most often at the edges of words.

Table 2 shows that filtering the segmentation results with the n-gram language model caused a significant improvement on the overall F-score for most languages, and led to significant changes in precision and recall. Whereas the original segmentation yielded balanced precision and recall (both 68%), the new filtering boosts precision to over 73%, with 64% recall. Which method is preferable (i.e. whether precision or recall is more important) is task-dependent.

In future work, we plan to draw on (Creutz and Lagus, 2006), who use a HMM with morphemic categories to impose morphotactic constraints. In such an approach, each element from the affix list is assigned with a certain probability to the underlying categories of "stem", "prefix" or "suffix", depending on the left and right perplexity of morphemes, as well as morpheme length and frequency. The transitional probabilities from one category to the next model the morphotactic rules of a language, which can thus be learnt automatically.

4.3 Learning Stem Variation

Stem variation through ablauting and umlauting (an English example is run–ran) is an interesting problem that cannot be captured by the algorithm outlined above, as variations take place within the morphemes. Stem variations can be context-dependent and do not constitute a morpheme in themselves. German umlauting and ablauting leads to data sparseness problems in morphological segmentation and affix acquisition. One problem is that affixes which usually cause ablauting or umlauting are very difficult to find. Typically, ablauted or umlauted stems are only seen with a very small number of different affixes, which means that the affix sets of such stems are divided into several unrelated subsets, causing the stem to be pruned from the stem candidate list. Secondly, ablauting and umlauting lead to low transitional probabilities at the positions in stems where these phenomena occur. Consider for example the affix set for the stem candidate *bockspr*, which contains the pseudoaffixes *ung*, *ünge* and *ingen*. The morphemes *sprung*, *sprüng* and *springen* are derived from the root *spring* 'to jump'. In the segmentation step this low transitional probability thus leads to oversegmentation.

We therefore investigated whether we can learn these regular stem variations automatically. A simple way to acquire the stem variations is to look at the suffix clusters which are calculated during the stem-acquisition step. When looking at the sets of substrings that are clustered together by having the same prefix, we found that they are often inflections of one another, because lexicalized compounds are used frequently in different inflectional variants. For example, we find *Trainingssprung* as well as *Trainingssprünge* in the corpus. The affix list of the stem candidate *trainings* thus contains the words *sprung* and *sprünge*. Edit distance can then be used to find differences between all words in a certain affix list. Pairs with small edit distances are stored and ranked by frequency. Regular transformation rules (e.g. ablauting and umlauting, $u \rightarrow \ddot{u}..e$) occur at the top of the list and are automatically accepted as rules (see Table 1). This method allows us to not only find the relation between two words in the lexicon (*Sprung* and *Sprünge*) but also to automatically learn rules that can be applied to unknown words to check whether their variant is a word in the lexicon.

freq.	diff.	examples
1682	a ä..e	sack-säcke, brach-bräche, stark-stärke
344	a ä	sahen-sähen, garten-gärten
321	u ü..e	flug-flüge, bund-bünde
289	ä a..s	verträge-vertrages, pässe-passes
189	o ö..e	chor-chöre, strom-ströme, ?röhre-rohr
175	t en	setzt-setzen, bringt-bringen
168	a u	laden-luden, *damm-dumm
160	ß ss	läßt-lässt, mißbrauch-missbrauch
[. . .]		
136	a en	firma-firmen, thema-themen
[. . .]		
2	ß g	*fließen-fliegen, *laßt-lagt
2	um o	*studiums-studios

Table 1: Excerpts from the stem variation detection algorithm results. Morphologically unrelated word pairs are marked with an asterisk.

We integrated information about stem variation from the regular stem transformation rules (those with the highest frequencies) into the segmentation step by creating equivalence sets of letters. For example, the rule $u \rightarrow ü..e$ generates an equivalence set {ü, u}. These two letters then count as the same letter when calculating transitional probabilities. We evaluated the benefit of integrating stem variation information for German on the German CELEX data set, and achieved an improvement of 2% in recall, without any loss in precision (F-measure: 69.4%, Precision: 68.1%, Recall: 70.8%; values for RePortS-stems). For better comparability to other systems and languages, results reported in the next section refer to the system version that does not incorporate stem variation.

5 Evaluation

For evaluating the different versions of the algorithm on English, Turkish and Finnish, we used the training and test sets from MorphoChallenge to enable comparison with other systems. Performance of the algorithm on German was evaluated on 244k manually annotated words from CELEX because German was not included in the MorphoChallenge data.

Table 2 shows that the introduction of the stem candidate acquisition step led to much higher recall on German, Finnish and Turkish, but caused some losses in precision. For English, adding both components did not have a large effect on either precision or recall. This means that this component is well behaved, i.e. it improves performance on languages where the intermediate stem-acquisition step is needed, but does not impair results on other languages. Recall for Finnish is still very low. It can be improved (at the expense of precision) by selecting the analysis with the largest number of segments in the segmentation step. The results for this heuristic was only evaluated on a smaller test set (ca. 700 wds), hence marked with an asterisk in Table 2.

Lang.	alg.version	F-Meas.	Prec.	Recall
Eng[1]	original	76.8%	76.2%	77.4%
	stems	67.6%	62.9%	73.1%
	n-gram seg.	75.1%	74.4%	75.9%
Ger[2]	original	59.2%	71.1%	50.7%
	stems	68.4%	68.1%	68.6%
	n-gram seg.	68.9%	73.7%	64.6%
Tur[1]	original	54.2%	72.9%	43.1%
	stems	61.8%	65.9%	58.2%
	n-gram seg.	64.2%	65.2%	63.3%
Fin[1]	original	47.1%	84.5%	32.6%
	stems	56.6%	74.1%	45.8%
	n-gram seg.	58.9%	76.1%	48.1%
	max-split*	61.3%	66.3%	56.9%

Table 2: Performance of the algorithm with the modifications on different languages.
[1]MorphoChallenge Data, [2]German CELEX

The algorithm is very efficient: When trained on the 240m tokens of the German TAZ corpus, it takes up less than 1 GB of memory. The training phase takes approx. 5 min on a 2.4GHz machine, and the segmentation of the 250k test words takes 3 min for the version that does the simple segmentation and about 8 min for the version that generates all possible segmentations and uses the language model.

5.1 Comparison to other systems

This modified version of the algorithm performs second best for English (after original RePortS) and ranks third for Turkish (after Bernhards algorithm with 65.3% F-measure and Morfessor-Categories-MAP with 70.7%). On German, our method significantly outperformed the other unsupervised algorithms, see Table 3. While most of the systems compared here were developed for languages other than German, (Bordag, 2006) describes a system initially built for German. When trained on the "Projekt Deutscher Wortschatz" corpus which comprises 24 million sentences, it achieves an F-score of 61% (precision 60%, recall 62%[2]) when evaluated on the full CELEX corpus.

[2]Data from personal communication.

morphology	F-Meas.	Prec.	Recall
SMOR-disamb2	83.6%	87.1%	80.4%
ETI	79.5%	75.4%	84.1%
SMOR-disamb1	71.8%	95.4%	57.6%
RePortS-lm	68.8%	73.7%	64.6%
RePortS-stems	68.4%	68.1%	68.6%
best Bernhard	63.5%	64.9%	62.1%
Bordag	61.4%	60.6%	62.3%
orig. RePortS	59.2%	71.1%	50.7%
best Morfessor 1.0	52.6%	70.9%	41.8%

Table 3: Evaluating rule-based and data-based systems for morphological segmentation with respect to CELEX manual morphological annotation.

Rule-based systems are currently the most common approach to morphological decomposition and perform better at segmenting words than state-of-the-art unsupervised algorithms (see Table 3 for performance of state-of-the-art rule-based systems evaluated on the same data). Both the ETI[3] and the SMOR (Schmid et al., 2004) systems rely on a large lexicon and a set of rules. The SMOR system returns a set of analyses that can be disambiguated in different ways. For details refer to pp. 29–33 in (Demberg, 2006).

5.2 Evaluation on Grapheme-to-Phoneme Conversion

Morphological segmentation is not of value in itself – the question is whether it can help improve results on an application. Performance improvements due to morphological information have been reported for example in MT, information retrieval, and speech recognition. For the latter task, morphological segmentations from the unsupervised systems presented here have been shown to improve accuracy (Kurimo et al., 2006).

Another motivation for evaluating the system on a task rather than on manually annotated data is that linguistically motivated morphological segmentation is not necessarily the best possible segmentation for a certain task. Evaluation against a manually annotated corpus prefers segmentations that are closest to linguistically motivated analyses. Furthermore, it might be important for a certain task to find a particular type of morpheme boundaries (e.g. boundaries between stems), but for another task it might be very important to find boundaries between stems and suffixes. The standard evaluation procedure does not differentiate between the types of mistakes made. Finally, only evaluation on a task can provide information as to whether high precision or high recall is more important, therefore, the decision as to which version of the algorithm should be chosen can only be taken given a specific task.

[3]Eloquent Technology, Inc. (ETI) TTS system. `www.mindspring.com/~ssshp/ssshp_cd/ss_eloq.htm`

morphology	F-Meas. (CELEX)	PER (dt)
CELEX	100%	2.64%
ETI	79.5%	2.78%
SMOR-disamb2	83.0%	3.00%
SMOR-disamb1	71.8%	3.28%
RePortS-lm	68.8%	3.45%
no morphology		3.63%
orig. RePortS	59.2%	3.83%
Bernhard	63.5%	3.88%
RePortS-stem	68.4%	3.98%
Morfessor 1.0	52.6%	4.10%
Bordag	64.1%	4.38%

Table 4: F-measure for evaluation on manually annotated CELEX and phoneme error rate (PER) from g2p conversion using a decision tree (dt).

For these reasons we decided to evaluate the segmentation from the new versions of the RePortS algorithm on a German grapheme-to-phoneme (g2p) conversion task. The evaluation on this task is motivated by the fact that (Demberg, 2007) showed that good-quality morphological preprocessing can improve g2p conversion results. We here compare the effect of using our system's segmentations to a range of different morphological segmentations from other systems. We ran each of the rule-based systems (ETI, SMOR-disamb1, SMOR-disamb2) and the unsupervised algorithms (original RePortS, Bernhard, Morfessor 1.0, Bordag) on the CELEX data set and retrained our decision tree (an implementation based on (Lucassen and Mercer, 1984)) on the different morphological segmentations.

Table 4 shows the F-score of the different systems when evaluated on the manually annotated CELEX data (full data set) and the phoneme error rate (PER) for the g2p conversion algorithm when annotated with morphological boundaries (smaller test set, since the decision tree is a supervised method and needs training data). As we can see from the results, the distribution of precision and recall (see Table 3) has an important impact on the conversion quality: the RePortS version with higher precision signifi-

cantly outperforms the other version on the task, although their F-measures are almost identical. Remarkably, the RePortS version that uses the filtering step is the only unsupervised system that beats the no-morphology baseline ($p < 0.0001$). While all other unsupervised systems tested here make the system perform worse than it would without morphological information, this new version improves accuracy on g2p conversion.

6 Conclusions

A significant improvement in F-score was achieved by three simple modifications to the RePortS algorithm: generating an intermediary high-precision stem candidate list, using a language model to disambiguate between alternative segmentations, and learning patterns for regular stem variation, which can then also be exploited for segmentation. These modifications improved results on four different languages considered: English, German, Turkish and Finnish, and achieved the best results reported so far for an unsupervised system for morphological segmentation on German. We showed that the new version of the algorithm is the only unsupervised system among the systems evaluated here that achieves sufficient quality to improve transcription performance on a grapheme-to-phoneme conversion task.

Acknowledgments

I would like to thank Emily Pitler and Samarth Keshava for making available the code of the RePortS algorithm, and Stefan Bordag and Delphine Bernhard for running their algorithms on the German data. Many thanks also to Matti Varjokallio for evaluating the data on the MorphoChallenge test sets for Finnish, Turkish and English. Furthermore, I am very grateful to Christoph Zwirello and Gregor Möhler for training the decision tree on the new morphological segmentation. I also want to thank Frank Keller and the ACL reviewers for valuable and insightful comments.

References

Delphine Bernhard. 2006. Unsupervised morphological segmentation based on segment predictability and word segments alignment. In *Proceedings of 2nd Pascal Challenges Workshop*, pages 19–24, Venice, Italy.

Stefan Bordag. 2006. Two-step approach to unsupervised morpheme segmentation. In *Proceedings of 2nd Pascal Challenges Workshop*, pages 25–29, Venice, Italy.

Mathias Creutz and Krista Lagus. 2006. Unsupervised models for morpheme segmentation and morphology learning. In *ACM Transaction on Speech and Language Processing*.

H. Déjean. 1998. Morphemes as necessary concepts for structures: Discovery from untagged corpora. In *Workshop on paradigms and Grounding in Natural Language Learning*, pages 295–299, Adelaide, Australia.

Vera Demberg. 2006. Letter-to-phoneme conversion for a German TTS-System. Master's thesis. *IMS, Univ. of Stuttgart*.

Vera Demberg. 2007. Phonological constraints and morphological preprocessing for grapheme-to-phoneme conversion. In *Proc. of ACL-2007*.

Eric Gaussier. 1999. Unsupervised learning of derivational morphology from inflectional lexicons. In *ACL '99 Workshop Proceedings*, University of Maryland.

CELEX German Linguistic User Guide, 1995. *Center for Lexical Information*. Max-Planck-Institut for Psycholinguistics, Nijmegen.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *computational Linguistics*, 27(2):153–198, June.

S. Goldwater and D. McClosky. 2005. Improving statistical mt through morphological analysis. In *Proc. of EMNLP*.

Margaret A. Hafer and Stephen F. Weiss. 1974. Word segmentation by letter successor varieties. *Information Storage and Retrieval 10*, pages 371–385.

Zellig Harris. 1955. From phoneme to morpheme. *Language 31*, pages 190–222.

Christian Jacquemin. 1997. Guessing morphology from terms and corpora. In *Research and Development in Information Retrieval*, pages 156–165.

S. Keshava and E. Pitler. 2006. A simpler, intuitive approach to morpheme induction. In *Proceedings of 2nd Pascal Challenges Workshop*, pages 31–35, Venice, Italy.

M. Kurimo, M. Creutz, M. Varjokallio, E. Arisoy, and M. Saraclar. 2006. Unsupervsied segmentation of words into morphemes – Challenge 2005: An introduction and evaluation report. In *Proc. of 2nd Pascal Challenges Workshop*, Italy.

J. Lucassen and R. Mercer. 1984. An information theoretic approach to the automatic determination of phonemic baseforms. In *ICASSP 9*.

C. Monson, A. Lavie, J. Carbonell, and L. Levin. 2004. Unsupervised induction of natural language morphology inflection classes. In *Proceedings of the Seventh Meeting of ACL-SIGPHON*, pages 52–61, Barcelona, Spain.

C. Monz and M. de Rijke. 2002. Shallow morphological analysis in monolingual information retrieval for Dutch, German, and Italian. In *Proceedings CLEF 2001, LNCS 2406.*

Sylvain Neuvel and Sean Fulop. 2002. Unsupervised learning of morphology without morphemes. In *Proc. of the Wshp on Morphological and Phonological Learning, ACL Pub.*

Helmut Schmid, Arne Fitschen, and Ulrich Heid. 2004. SMOR: A German computational morphology covering derivation, composition and inflection. In *Proc. of LREC.*

Patrick Schone and Daniel Jurafsky. 2000. Knowledge-free induction of morphology using latent semantic analysis. In *Proc. of CoNLL-2000 and LLL-2000*, Lisbon, Portugal.

Tageszeitung (TAZ) Corpus. Contrapress Media GmbH. `https://www.taz.de/pt/.etc/nf/dvd`.

David Yarowski and Richard Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of ACL 2000*, Hong Kong.

Using Mazurkiewicz Trace Languages for Partition-Based Morphology

François Barthélemy
CNAM Cedric, 292 rue Saint-Martin, 75003 Paris (France)
INRIA Atoll, domaine de Voluceau, 78153 Le Chesnay cedex (France)
barthe@cnam.fr

Abstract

Partition-based morphology is an approach of finite-state morphology where a grammar describes a special kind of regular relations, which split all the strings of a given tuple into the same number of substrings. They are compiled in finite-state machines. In this paper, we address the question of merging grammars using different partitionings into a single finite-state machine. A morphological description may then be obtained by parallel or sequential application of constraints expressed on different partition notions (e.g. morpheme, phoneme, grapheme). The theory of Mazurkiewicz Trace Languages, a well known semantics of parallel systems, provides a way of representing and compiling such a description.

1 Partition-Based Morphology

Finite-State Morphology is based on the idea that regular relations are an appropriate formalism to describe the morphology of a natural language. Such a relation is a set of pairs, the first component being an actual form called *surface form*, the second component being an abstract description of this form called *lexical form*. It is usually implemented by a finite-state transducer. Relations are not oriented, so the same transducer may be used both for analysis and generation. They may be non-deterministic, when the same form belongs to several pairs. Furthermore, finite state machines have interesting properties, they are composable and efficient.

There are two main trends in Finite-State Morphology: rewrite-rule systems and two-level rule systems. Rewrite-rule systems describe the morphology of languages using contextual rewrite rules which are easily applied in cascade. Rules are compiled into finite-state transducers and merged using transducer composition (Kaplan and Kay, 1994).

The other important trend of Finite-State Morphology is Two-Level Morphology (Koskenniemi, 1983). In this approach, not only pairs of lexical and surface strings are related, but there is a one-to-one correspondence between their symbols. It means that the two strings of a given pair must have the same length. Whenever a symbol of one side does not have an actual counterpart in the other string, a special symbol 0 is inserted at the relevant position in order to fulfill the same-length constraint. For example, the correspondence between the surface form `spies` and the morpheme concatenation `spy+s` is given as follows:

s	p	y	0	+	s
s	p	i	e	0	s

Same-length relations are closed under intersection, so two-level grammars describe a system as the simultaneous application of local constraints.

A third approach, Partition-Based Morphology, consists in splitting the strings of a pair into the same number of substrings. The same-length constraint does not hold on symbols but on substrings. For example, `spies` and `spy+s` may be partitioned as follows:

s	p	y	+	s
s	p	ie	ϵ	s

The partition-based approach was first proposed by (Black et al., 1987) and further improved by (Pulman and Hepple, 1993) and (Grimley-Evans et al.,

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 928–935,
Prague, Czech Republic, June 2007.

1996). It has been used to describe the morphology of Syriac (Kiraz, 2000), Akkadian (Barthélemy, 2006) and Arabic Dialects (Habash et al., 2005). These works use multi-tape transducers instead of usual two tape transducers, describing a special case of n-ary relations instead of binary relations.

Definition 1 *Partitioned n-relation*
A partitioned n-relation is a set of finite sequences of string n-tuples.

For instance, the n-tuple sequence of the example $(spy, spies)$ given above is $(s,s)(p,p)(y,ie)(+,\epsilon)(s,s)$. Of course, all the partitioned n-relations are not recognizable using a finite-state machine. Grimley-Evans and al. propose a partition-based formalism with a strong restriction: the string n-tuples used in the sequences belong to a finite set of such n-tuples (the centers of context-restriction rules). They describe an algorithm which compiles a set of contextual rules describing a partitioned n-relation into an epsilon-free letter transducer. (Barthélemy, 2005) proposed a more powerful framework, where the relations are defined by concatenating tuples of independent regular expressions and operations on partitioned n-relations such as intersection and complementation are considered.

In this paper, we propose to use Mazurkiewicz Trace Languages instead of partitioned relation as the semantics of partition-based morphological formalisms. The benefits are twofold: firstly, there is an extension of the formal power which allows the combination of morphological description using different partitionings of forms. Secondly, the compilation of such languages into finite-state machines has been exhaustively studied. Their closure properties provide operations useful for morphological purposes.

They include the concatenation (for instance for compound words), the intersection used to merge local constraints, the union (modular lexicon), the composition (cascading descriptions, form recognition and generation), the projection (to extract one level of the relation), the complementation and set difference, used to compile contextual rules following the algorithms in (Kaplan and Kay, 1994), (Grimley-Evans et al., 1996) and (Yli-Jyrä and Koskenniemi, 2004).

The use of the new semantics does not imply any change of the user-level formalisms, thanks to a straightforward homomorphism from partitioned n-relations to Mazurkiewicz Trace Languages.

2 Mazurkiewicz Trace Languages

Within a given n-tuple, there is no meaningful order between symbols of the different levels. Mazurkiewicz trace languages is a theory which expresses partial ordering between symbols. They have been defined and studied in the realm of parallel computing. In this section, we recall their definition and some classical results. (Diekert and Métivier, 1997) gives an exhaustive presentation on the subject with a detailed bibliography. It contains all the results mentioned here and refers to their original publication.

2.1 Definitions

A Partially Commutative Monoid is defined on an alphabet Σ with an independence binary relation I over $\Sigma \times \Sigma$ which is symmetric and irreflexive. Two independent symbols commute freely whereas non-independent symbols do not. I defines an equivalence relation $\sim_I$ on Σ^*: two words are equivalent if one is the result of a series of commutation of pairs of successive symbols which belong to I. The notation $[x]$ is used to denote the equivalence class of a string x with respect to $\sim_I$.

The Partially Commutative Monoid $M(\Sigma, I)$ is the quotient of the free monoid Σ^* by the equivalence relation $\sim_I$.

The binary relation $D = (\Sigma \times \Sigma) - I$ is called the dependence relation. It is reflexive and symmetric.

φ is the canonical homomorphism defined by:

$$\begin{array}{rccc} \varphi: & \Sigma^* & \rightarrow & M(\Sigma, I) \\ & x & \mapsto & [x] \end{array}$$

A Mazurkiewicz trace language (abbreviation: trace language) is a subset of a partially commutative monoid $M(\Sigma, I)$.

2.2 Recognizable Trace Languages

A trace language T is said *recognizable* if there exists an homomorphism ν from $M(\Sigma, I)$ to a finite monoid S such that $T = \nu^{-1}(F)$ for some $F \subseteq S$. A recognizable Trace Language may be implemented by a Finite-State Automaton.

A trace $[x]$ is said to be connected if the dependence relation restricted to the alphabet of $[x]$ is a connected graph. A trace language is connected if all its traces are connected.

A string x is said to be in lexicographic normal form if x is the smallest string of its equivalence class $[x]$ with respect to the lexicographic ordering induced by an ordering on Σ. The set of strings in lexicographic normal form is written $LexNF$. This set is a regular language which is described by the following regular expression:
$LexNF = \Sigma^* - \bigcup_{(a,b)\in I, a<b} \Sigma^* b (I(a))^* a \Sigma^*$
where $I(a)$ denotes the set of symbols independent from a.

Property 1 *Let $T \subseteq M(\Sigma, I)$ be a trace language. The following assertions are equivalent:*

- *T is recognizable*
- *T is expressible as a rational expression where the Kleene star is used only on connected languages.*
- *The set $Min(T) = \{x \in LexNF | [x] \in T\}$ is a regular language over Σ^*.*

Recognizability is closely related to the notion of iterative factor, which is the language-level equivalent of a loop in a finite-state machine. If two symbols a and b such that $a < b$ belong to a loop, and if the loop is traversed several times, then occurrences of a and b are interlaced. For such a string to be in lexicographic normal form, a dependent symbol must appear in the loop between b and a.

2.3 Operations and closure properties

Recognizable trace languages are closed under intersection and union. Furthermore, $\text{Min}(T_1) \cup \text{Min}(T_2) = \text{Min}(T_1 \cup T_2)$ and $\text{Min}(T_1) \cap \text{Min}(T_2) = \text{Min}(T_1 \cap T_2)$. It comes from the fact that intersection and union do not create new iterative factor. The property on lexicographic normal form comes from the fact that all the traces in the result of the operation belong to at least one of the operands which are in normal form.

Recognizable trace language are closed under concatenation. Concatenation do not create new iterative factors. The concatenation $\text{Min}(T_1)\text{Min}(T_2)$ is not necessarily in lexicographic normal form. For instance, suppose that $a > b$. Then $\{[a]\}.\{[b]\} = \{[ab]\}$, but $\text{Min}(\{[a]\}) = a, \text{Min}(\{[b]\}) = b$, and $\text{Min}(\{[ab]\}) = ba$.

Recognizable trace languages are closed under complementation.

Recognizable Trace Languages are not closed under Kleene star. For instance, $a < b$, $\text{Min}([ab]^*) = a^n b^n$ which is known not to be regular.

The projection on a subset S of Σ is the operation written π_S, which deletes all the occurrences of symbols in $\Sigma - S$ from the traces. Recognizable trace languages are not closed under projection. The reason is that the projection may delete symbols which makes the languages of loops connected.

3 Partitioned relations and trace languages

It is possible to convert a partitioned relation into a trace language as follows:

- represent the partition boundaries using a symbol ω not in Σ.
- distinguish the symbols according to the component (tape) of the n-tuple they belong to. For this purpose, we will use a subscript.
- define the dependence relation D by:
  - ω is dependent from all the other symbols
  - symbols in Σ sharing the same subscript are mutually dependent whereas symbols having different subscript are mutually independent.

For instance, the *spy* n-tuple sequence $(s,s)(p,p)(y,ie)(+,\epsilon)(s,s)$ is translated into the trace $\omega s_1 s_2 \omega p_1 p_2 \omega y_1 i_2 e_2 \omega +_1 \omega s_1 s_2 \omega$. The figure 1 gives the partial order between symbols of this trace.

The dependence relation is intuitively sound. For instance, in the third n-tuple, there is a dependency between i and e which cannot be permuted, but there is no dependency between i (resp. e) and y: i is neither before nor after y. There are three equivalent permutations: $y_1 i_2 e_2$, $i_2 y_1 e_2$ and $i_2 e_2 y_1$. In an implementation, one canonical representation must be chosen, in order to ensure that set operations, such as intersection, are correct. The notion of lexicographic normal form, based on any arbitrary but fixed order on symbols, gives such a canonical form.

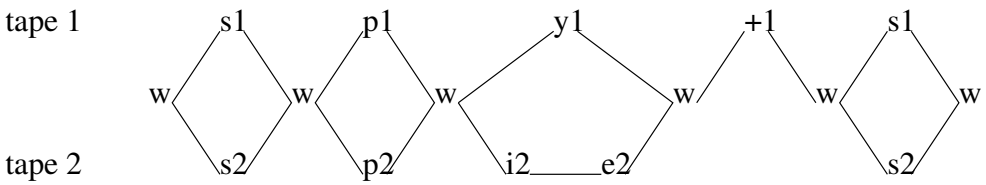


Figure 1: Partially ordered symbols

The compilation of the trace language into a finite-state automaton has been studied through the notion of recognizability. This automaton is very similar to an n-tape transducer. The Trace Language theory gives properties such as closure under intersection and soundness of the lexicographic normal form, which do not hold for usual transducers classes. It also provides a criterion to restrict the description of languages through regular expressions. This restriction is that the closure operator (Kleene star) must occur on connected languages only. In the translation of a partition-based regular expression, a star may appear either on a string of symbols of a given tape or on a string with at least one occurrence of ω.

Another benefit of Mazurkiewicz trace languages with respect to partitioned relations is their ability to represent the segmentation of the same form using two different partitionings. The example of figure 2 uses two partitionings of the form `spy+s`, one based on the notion of morpheme, the other on the notion of phoneme. The notation `<pos=noun>` and `<number=pl>` stands for two single symbols. Flat feature structures over (small) finite domains are easily represented by a string of such symbols. N-tuples are not very convenient to represent such a system.

Partition-based formalism are especially adapted to express relations between different representation such as feature structures and affixes, with respect to two-level morphology which imposes an artificial symbol-to-symbol mapping.

A multi-partitioned relation may be obtained by merging the translation of two partition-based grammars which share one or more common tapes. Such a merging is performed by the join operator of the relational algebra. Using a partition-based grammar for recognition or generation implies such an operation: the grammar is joined with a 1-tape machine without partitioning representing the form to be recognized (surface level) or generated (lexical level).

4 Multi-Tape Trace Languages

In this section, we define a subclass of Mazurkiewicz Trace Languages especially adapted to partition-based morphology, thanks to an explicit notion of tape partially synchronized by partition boundaries.

Definition 2 *A multi-tape partially commutative monoid is defined by a tuple* $(\Sigma, \Theta, \Omega, \mu)$ *where*

- Σ *is a finite set of symbols called the alphabet.*
- Θ *is a finite set of symbols called the tapes.*
- Ω *is a finite set of symbols which do not belong to* Σ*, called the partition boundaries.*
- μ *is a mapping from* $\Sigma \cup \Omega$ *to* 2^θ *such that* $\mu(x)$ *is a singleton for any* $x \in \Sigma$*.*

It is the Partially Commutative Monoid $M(\Sigma \cup \Omega, I_\mu)$ *where the independence relation is defined by* $I_\mu = \{(x, y) \in \Sigma \cup \Omega \times \Sigma \cup \Omega | \mu(x) \cap \mu(y) = \emptyset\}$. *Notation:* $MPM(\Sigma, \Theta, \Omega, \mu)$.

A Multi-Tape Trace Language is a subset of a Multi-Tape partially commutative monoid.

We now address the problem of relational operations over Recognizable Multi-Tape Trace Languages. Recognizable languages may be implemented by finite-state automata in lexicographic normal form, using the morphism φ^{-1}. Operations on trace languages are implemented by operations on finite-state automata. We are looking for implementations preserving the normal form property, because changing the order in regular languages is not a standard operation.

Some set operations are very simple to implement, namely union, intersection and difference.

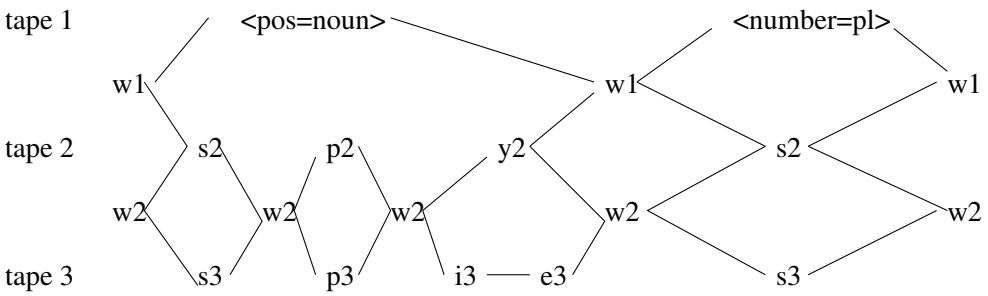


Figure 2: Two partitions of the same tape

The elements of the result of such an operation belongs to one or both operands, and are therefore in lexicographic normal form. If we write $Min(T)$ the set $Min(T) = \{x \in LexNF | [x] \in T\}$, where T is a Multi-Tape Trace Language, we have trivially the properties:

- $Min(T_1 \cup T_2) = Min(T_1) \cup Min(T_2)$
- $Min(T_1 \cap T_2) = Min(T_1) \cap Min(T_2)$
- $Min(T_1 - T_2) = Min(T_1) - Min(T_2)$

Implementing the complementation is not so straightforward because $Min(\overline{T})$ is usually not equal to $\overline{Min(T)}$. The later set contains strings not in lexical normal forms which may belong to the equivalence class of a member of T with respect to $\sim_I$. The complementation must not be computed with respect to regular languages but to LexNF.
$Min(\overline{T}) = LexNF - Min(T)$

As already mentioned, the concatenation of two regular languages in lexicographic normal form is not necessarily in normal form. We do not have a general solution to the problem but two partial solutions. Firstly, it is easy to test whether the result is actually in normal form or not. Secondly, the result is in normal form whenever a synchronization point belonging to all the levels is inserted between the strings of the two languages. Let $\omega_u \in \Omega, \mu(\omega_u) = \Theta$. Then, $Min(T_1.\{\omega_u\}.T_2) = Min(T_1).Min(\omega_u).Min(T_2)$.

The closure (Kleene star) operation creates a new iterative factor and therefore, the result may be a non recognizable trace language. Here again, concatenating a global synchronization point at the end of the language gives a trace language closed under Kleene star. By definition, such a language is connected. Furthermore, the result is in normal form.

So far, operations have operands and the result belonging to the same Multi-tape Monoid. It is not the case of the last two operations: projection and join.

We use the the operators Dom, Range, and the relations Id and Insert as defined in (Kaplan and Kay, 1994):

- Dom$(R) = \{x | \exists y, (x, y) \in R\}$
- Range$(R) = \{y | \exists x, (x, y) \in R\}$
- Id$(L) = \{(x, x) | x \in L\}$
- Insert$(S) = (\text{Id}(\Sigma) \cup (\{\epsilon\} \times S))^*$. It is used to insert freely symbols from S in a string from Σ^*. Conversely, Insert$(S)^{-1}$ removes all the occurrences of symbols from S, if $S \cap \Sigma = \emptyset$.

The result of a projection operation may not be recognizable if it deletes symbols making iterative factors connected. Furthermore, when the result is recognizable, the projection on $Min(T)$ is not necessarily in normal form. Both phenomena come from the deletion of synchronization points. Therefore, a projection which deletes only symbols from Σ is safe. The deletion of synchronization points is also possible whenever they do not synchronize anything more in the result of the projection because all but possibly one of its tapes have been deleted.

In the tape-oriented computation system, we are mainly interested in the projection which deletes some tapes and possibly some related synchronization points.

Property 2 *Projection*
Let T be a trace language over the MTM $M = (\Sigma, \Theta, w, \mu)$. Let $\Omega_1 \subset \Omega$ and $\Theta_1 \subset \Theta$. If

$\forall \omega \in \Omega - \Omega_1, |\mu(\omega) \cap \Theta_1| \leq 1$, *then* $Min(\pi_{\Theta_1,\Omega_1}(T)) = Range(Insert(\{x \in \Sigma | \mu(x) \notin \Theta_1\} \cup \Omega - \Omega_1)^{-1} \circ Min(T))$

The join operation is named by analogy with the operator of the relational algebra. It has been defined on finite-state transducers (Kempe et al., 2004).

Definition 3 *Multi-tape join*
Let $T_1 \subset MTM(\Sigma_1, \Theta_1, \Omega_1, \mu_1)$ *and* $T_2 \subset TM(\Sigma_2, \Theta_2, \Omega_2, \mu_2)$ *be two multi-tape trace languages.* $T_1 \bowtie T_2$ *is defined if and only if*

- $\forall \sigma \in \Sigma_1 \cap \Sigma_2, \mu_1(\sigma) \cap \Theta_2 = \mu_2(\sigma) \cap \Theta_1$
- $\forall \omega \in \Omega_1 \cap \Omega_2, \mu_1(\omega) \cap \Theta_2 = \mu_2(\omega) \cap \Theta_1$

The Multi-tape Trace Language $T_1 \bowtie T_2$ *is defined on the Multi-tape Partially Commutative Monoid* $MTM(\Sigma_1 \cup \Sigma_2, \Theta_1 \cup \Theta_2, \Omega_1 \cup \Omega_2, \mu)$ *where* $\mu(x) = \mu_1(x) \cup \mu_2(x)$. *It is defined by* $\pi_{\Sigma_1 \cup \Theta_1 \cup \Omega_1}(T_1 \bowtie T_2) = T_1$ *and* $\pi_{\Sigma_2 \cup \Theta_2 \cup \Omega_2}(T_1 \bowtie T_2) = T_2$.

If the two operands T_1 and T_2 belong to the same MTM, then $T_1 \bowtie T_2 = T_1 \cap T_2$. If the operands belong to disjoint monoids (which do not share any symbol), then the join is a Cartesian product.

The implementation of the join relies on the finite-state intersection algorithm. This algorithm works whenever the common symbols of the two languages appear in the same order in the two operands. The normal form does not ensure this property, because symbols in the common part of the join may be synchronized by tapes not in the common part, by transitivity, like in the example of the figure 3. In this example, `c` on tape 3 and `f` on tape 1 are ordered $c < f$ by transitivity using tape 2.

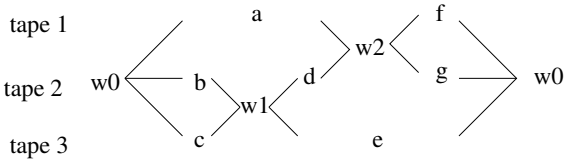


Figure 3: indirect tape synchronization

Let $T \subseteq MPM(\Sigma, \Theta, \Omega, \mu)$ a multi-partition trace language. Let G_T be the labeled graph where the nodes are the tape symbols from Θ and the edges are the set $\{(x, \omega, y) \in \Theta \times \Omega \times \Theta | x \in \mu(\omega)$ and $y \in \mu(\omega)\}$. Let Sync(Θ) be the set defined by Sync(Θ) = $\{\omega \in \Omega | \omega$ appears in G_T on a path between two tapes of $\Theta\}$.

The G_T graph for example of the figure 3 is given in figure 4 and Sync($\{1, 3\}$) = $\{\omega_0, \omega_1, \omega_2\}$.

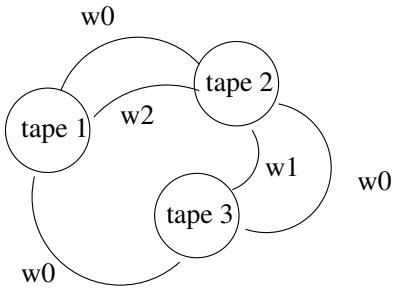


Figure 4: the G_T graph

Sync(Θ) is different from $\mu^{-1}(\Theta) \cap \Omega$ because some synchronization points may induce an order between two tapes by transitivity, using other tapes.

Property 3 *Let* $T_1 \subseteq MPM(\Sigma_1, \Theta_1, \Omega_1, \mu_1)$ *and* $T_2 \subseteq MPM(\Sigma_2, \Theta_2, \Omega_2, \mu_2)$ *be two multi-partition trace languages. Let* $\Sigma = \Sigma_1 \cap \Sigma_2$ *and* $\Omega = \Omega_1 \cap \Omega_2$. *If Sync*$(\Theta_1 \cap \Theta_2) \subseteq \Omega$, *then* $\pi_{\Sigma \cup \Omega}(Min(T_1)) \cap \pi_{\Sigma \cup \Omega}(Min(T_2)) = Min(\pi_{\Sigma \cup \Omega}(T_1) \cap \pi_{\Sigma \cup \Omega}(T_2)$

This property expresses the fact that symbols belonging to both languages appear in the same order in lexicographic normal forms whenever all the direct and indirect synchronization symbols belong to the two languages too.

Property 4 *Let* $T_1 \subseteq MPM(\Sigma_1, \Theta_1, \Omega_1, \mu_1)$ *and* $T_2 \subseteq MPM(\Sigma_2, \Theta_2, \Omega_2, \mu_2)$ *be two multi-partition trace languages. If* $\Theta_1 \cap \Theta_2$ *is a singleton* $\{\theta\}$ *and if* $\forall \omega \in \Omega_1 \cap \Omega_2, \theta \in \mu(\omega)$, *then* $\pi_{\Sigma \cup \Omega}(Min(T_1)) \cap \pi_{\Sigma \cup \Omega}(Min(T_2)) = Min(\pi_{\Sigma \cup \Omega}(T_1) \cap \pi_{\Sigma \cup \Omega}(T_2)$

This second property expresses the fact that symbols appear necessarily in the same order in the two operands if the intersection of the two languages is restricted to symbols of a single tape. This property is straightforward since symbols of a given tape are mutually dependent.

We now define a computation over $(\Sigma \cup \Omega)^*$ which computes $Min(T_1 \bowtie T_2)$.

Let $T_1 \subset MTM(\Sigma_1, \Theta_1, \omega_1, \mu_1)$ and $T_2 \subset MTM(\Sigma_2, \Theta_2, \Omega_2, \mu_2)$ be two recognizable multi-tape trace languages.
If Sync($\Theta_1 \cap \Theta_2$) $\subseteq \Omega$, then $Min(T_1 \bowtie T_2) =$ Range($Min(T_1 \circ$ Insert($\Sigma_2 - \Sigma_1) \circ$ Id(LexNF)) $\cap$ Range($Min(T_2) \circ$ Insert($\Sigma_1 - \Sigma_2) \circ$ Id(LexNF)).

5 A short example

We have written a morphological description of Turkish verbal morphology using two different partitionings. The first one corresponds to the notion of affix (morpheme). It is used to describe the morphotactics of the language using rules such as the following context-restriction rule:

```
(y?I4m,1 sing) ⇒
    (I?yor,prog)|(y?E2cE2k,future) __
```

In this rule, $y^?$ stands for an optional y, I^4 and E^2 for abstract vowels which realizations are subject to vowel harmony and $I^?$ is an optional occurrence of the first vowel. The rule may be read: the suffix $y^?I^4m$ denoting a first person singular may appear only after the suffix of progressive or the suffix of future[1]. Such rules describe simply affix order in verbal forms.

The second partitioning is a symbol-to-symbol correspondence similar to the one used in standard two-level morphology. This partitioning is more convenient to express the constraints of vowel harmony which occurs anywhere in the affixes and does not depend on affix boundaries.

Here are two of the rules implementing vowel harmony:

```
(I4,i) ⇒ (Vow,e|i) (Cons,Cons)* __
(I4,u) ⇒ (Vow,o|u) (Cons,Cons)* __
```

Vow and Cons denote respectively the sets of vowels and consonants. These rules may be read: *a symbol I^4 is realized as i (resp. u) whenever the closest preceding vowel is realized as e or i (resp. o or u).*

The realization or not of an optional letter may be expressed using one or the other partitioning. These optional letters always appear in the first position of an affix and depends only on the last letter of the preceding affix.

```
(y?,y) ⇒ (Vow,Vow) __
```

Here is an example of a verbal form given as a 3-tape relation partitioned using the two partitionings.

verbal root			prog				1 sing		
g	e	l	$I^?$	y	o	r	$Y^?$	I^4	m
g	e	l	i	y	o	r	ϵ	u	m

The translation of each rule into a Multi-tape Trace Language involves two tasks: introducing partition boundary symbols at each frontier between partitions. A different symbol is used for each kind of partitioning. Distinguishing symbols from different tapes in order to ensure that $\mu(x)$ is a singleton for each $x \in \Sigma$. Symbols of Σ are therefore pairs with the symbol appearing in the rule as first component and the tape identifier, a number, as second component.

Any complete order between symbols would define a lexicographic normal form. The order used by our system orders symbol with respect to tapes: symbols of the first tape are smaller than the symbols of tape 2, and so on. The order between symbols of a same tape is not important because these symbols are mutually dependent. The translation of a tuple $(a_1 \dots a_n, b_1 \dots b_m)$ is $(a_1, 1) \dots (a_n, 1)(b_1, 2) \dots (b_m, 2)\omega_1$. Such a string is in lexicographic normal form. Furthermore, this expression is connected, thanks to the partition boundary which synchronizes all the tapes, so its closure is recognizable. The concatenation too is safe.

All contextual rules are compiled following the algorithm in (Yli-Jyrä and Koskenniemi, 2004) [2]. Then all the rules describing affixes are intersected in an automaton, and all the rules describing surface transformation are intersected in another automaton. Then a join is performed to obtain the final machine. This join is possible because the intersection of the two languages consists in one tape (cf. property 4). Using it either for recognition or generation is also done by a join, possibly followed by a projection.

For instance, to recognize a surface form $geliyorum$, first compile it in the multi-tape trace language $(g, 3)(e, 3)(l, 3) \dots (m, 3)$, join it with the morphological description, and then project the result on tape 1 to obtain an abstract form `(verbal root,1)(prog,1)(1 sing,1)`. Finally extract the first component of each pair.

6 Conclusion

Partition-oriented rules are a convenient way to describe some of the constraints involved in the morphology of the language, but not all the constraints refer to the same partition notion. Describing a rule

[1] The actual rule has 5 other alternative tenses. It has been shortened for clarity.

[2] Two other compilation algorithm also work on the rules of this example (Kaplan and Kay, 1994), (Grimley-Evans et al., 1996). (Yli-Jyrä and Koskenniemi, 2004) is more general.

with an irrelevant one is sometimes difficult and inelegant. For instance, describing vowel harmony using a partitioning based on morphemes takes necessarily several rules corresponding to the cases where the harmony is within a morpheme or across several morphemes.

Previous partition-based formalisms use a unique partitioning which is used in all the contextual rules. Our proposition is to use several partitionings in order to express constraints with the proper granularity. Typically, these partitionings correspond to the notions of morphemes, phonemes and graphemes.

Partition-based grammars have the same theoretical power as two-level morphology, which is the power of regular languages. It was designed to remain finite-state and closed under intersection. It is compiled in finite-state automata which are formally equivalent to the epsilon-free letter transducers used by two-level morphology. It is simply more easy to use in some cases, just like two-level rules are more convenient than simple regular expressions for some applications.

Partition-Based morphology is convenient whenever the different levels use very different representations, like feature structures and strings, or different writing systems (e.g. Japanese hiragana and transcription). Two-level rules on the other hand are convenient whenever the related strings are variants of the same representation like in the example (spy+s,spies). Note that multi-partition morphology may use a one-to-one correspondence as one of its partitionings, and therefore is compatible with usual two-level morphology.

With respect to rewrite rule systems, partition-based morphology gives better support to parallel rule application and context definition may involve several levels. The counterpart is a risk of conflicts between contextual rules.

Acknowledgement

We would like to thank an anonymous referee of this paper for his/her helpful comments.

References

François Barthélemy. 2005. Partitioning multitape transducers. In *International Workshop on Finite State Methods in Natural Language Processing (FSMNLP)*, Helsinki, Finland.

François Barthélemy. 2006. Un analyseur morphologique utilisant la jointure. In *Traitement Automatique de la Langue Naturelle (TALN)*, Leuven, Belgium.

Alan Black, Graeme Ritchie, Steve Pulman, and Graham Russell. 1987. Formalisms for morphographemic description. In *Proceedings of the third conference on European chapter of the Association for Computational Linguistics (EACL)*, pages 11–18.

Volker Diekert and Yves Métivier. 1997. Partial commutation and traces. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Vol. 3*, pages 457–534. Springer-Verlag, Berlin.

Edmund Grimley-Evans, George Kiraz, and Stephen Pulman. 1996. Compiling a partition-based two-level formalism. In *COLING*, pages 454–459, Copenhagen, Denmark.

Nizar Habash, Owen Rambow, and George Kiraz. 2005. Morphological analysis and generation for arabic dialects. In *Proceedings of the ACL Workshop on Semitic Languages*, Ann Harbour, Michigan.

Ronald M. Kaplan and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20:3:331–378.

André Kempe, Jean-Marc Champarnaud, and Jason Eisner. 2004. A note on join and auto-intersection of *n*-ary rational relations. In B. Watson and L. Cleophas, editors, *Proc. Eindhoven FASTAR Days*, Eindhoven, Netherlands.

George Anton Kiraz. 2000. Multitiered nonlinear morphology using multitape finite automata: a case study on syriac and arabic. *Comput. Linguist.*, 26(1):77–105.

Kimmo Koskenniemi. 1983. Two-level model for morphological analysis. In *IJCAI-83*, pages 683–685, Karlsruhe, Germany.

Stephen G. Pulman and Mark R. Hepple. 1993. A feature-based formalism for two-level phonology. *Computer Speech and Language*, 7:333–358.

Anssi Yli-Jyrä and Kimmo Koskenniemi. 2004. Compiling contextual restrictions on strings into finite-state automata. In B. Watson and L. Cleophas, editors, *Proc. Eindhoven FASTAR Days*, Eindhoven, Netherlands.

Much ado about nothing: A social network model of Russian paradigmatic gaps

Robert Daland **Andrea D. Sims** **Janet Pierrehumbert**

Department of Linguistics
Northwestern University
2016 Sheridan Road
Evanston, IL 60208 USA
r-daland, andrea-sims, jbp@northwestern.edu

Abstract

A number of Russian verbs lack 1sg non-past forms. These paradigmatic gaps are puzzling because they seemingly contradict the highly productive nature of inflectional systems. We model the persistence and spread of Russian gaps via a multi-agent model with Bayesian learning. We ran three simulations: no grammar learning, learning with arbitrary analogical pressure, and morphophonologically conditioned learning. We compare the results to the attested historical development of the gaps. Contradicting previous accounts, we propose that the persistence of gaps can be explained in the absence of synchronic competition between forms.

1 Introduction

Paradigmatic gaps present an interesting challenge for theories of inflectional structure and language learning. Wug tests, analogical change and children's overextensions of regular patterns demonstrate that inflectional morphology is highly productive. Yet lemmas sometimes have "missing" inflected forms. For example, in Russian the majority of verbs have first person singular (1sg) non-past forms (e.g., *posadit'* 'to plant', *posažu* 'I will plant'), but no 1sg form for a number of similar verbs (e.g., *pobedit'* 'to win', **pobežu* 'I will win'). The challenge lies in explaining this apparent contradiction. Given the highly productive nature of inflection, why do paradigmatic gaps arise? Why do they persist?

One approach explains paradigmatic gaps as a problem in generating an acceptable form. Under this hypothesis, gaps result from irreconcilable conflict between two or more inflectional patterns. For example, Albright (2003) presents an analysis of Spanish verbal gaps based on the Minimal Generalization Learner (Albright and Hayes 2002). In his account, competition between mid-vowel diphthongization (e.g., *s[e]ntir* 'to feel', *s[je]nto* 'I feel') and non-diphthongization (e.g., *p[e]dir* 'to ask', *p[i]do* 'I ask') leads to paradigmatic gaps in lexemes for which the applicability of diphthongization has low reliability (e.g., *abolir* 'to abolish, **ab[we]lo*, **ab[o]lo* 'I abolish').

However, this approach both overpredicts and underpredicts the existence of gaps cross-linguistically. First, it predicts that gaps should occur whenever the analogical forces determining word forms are contradictory and evenly weighted. However, variation between two inflectional patterns seems to more commonly result from such a scenario. Second, the model predicts that if the form-based conflict disappears, the gaps should also disappear. However, in Russian and probably in other languages, gaps persist even after the loss of competing inflectional patterns or other synchronic form-based motivation (Sims 2006).

By contrast, our approach operates at the level of inflectional property sets (IPS), or more properly, at the level of inflectional paradigms. We propose that once gaps are established in a language for whatever reason, they persist because learners infer the relative non-use of a given

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 936–943,
Prague, Czech Republic, June 2007.

combination of stem and IPS.[1] Put differently, we hypothesize that speakers possess at least two kinds of knowledge about inflectional structure: (1) knowledge of how to generate the appropriate form for a given lemma and IPS, and (2) knowledge of the probability with which that combination of lemma and property set is expressed, regardless of the form. Our approach differs from previous accounts in that persistence of gaps is attributed to the latter kind of knowledge, and does not depend on synchronic morphological competition.

We present a case study of the Russian verbal gaps, which are notable for their persistence. They arose between the mid 19th and early 20th century (Baerman 2007), and are still strongly attested in the modern language, but have no apparent synchronic morphological cause.

We model the persistence and spread of the Russian verbal gaps with a multi-agent model with Bayesian learning. Our model has two kinds of agents, adults and children. A model cycle consists of two phases: a production-perception phase, and a learning-maturation phase. In the production-perception phase, adults produce a batch of linguistic data (verb forms), and children listen to the productions from the adults they know. In the learning-maturation phase, children build a grammar based on the input they have received, then mature into adults. The existing adults die off, and the next generation of children is born.

Our model exhibits similar behavior to what is known about the development of Russian gaps.

2 The historical and distributional facts of Russian verbal gaps

2.1 Traditional descriptions

Grammars and dictionaries of Russian frequently cite paradigmatic gaps in the 1sg non-past. Nine major dictionaries and grammars, including Švedova (1982) and Zaliznjak (1977), yielded a combined list of 96 gaps representing 68 distinct stems. These verbal gaps fall almost entirely into the second conjugation class, and they overwhelmingly affect the subgroup of dental stems. Commonly cited gaps include: **galžu* 'I make a hubbub'; **očučus'* 'I come to be (REFL)'; 1SG **oščušču* 'I feel'; **pobežu* 'I will win'; and **ubežu* 'I will convince'.[2]

There is no satisfactory synchronic reason for the existence of the gaps. The grouping of gaps among 2nd conjugation dental stems is seemingly non-arbitrary because these are exactly the forms that would be subject to a palatalizing morphophonological alternation ($t^j \rightarrow tʃ$ or $ʃ^j$, $d^j \rightarrow ʒ$, $s^j \rightarrow ʃ$, $z^j \rightarrow ʒ$). Yet the Russian gaps do not meet the criteria for morphophonological competition as intended by Albright's (2003) model, because the alternations apply automatically in Contemporary Standard Russian. Analogical forces should thus heavily favor a single form, for example, *pobežu*.

Traditional explanations for the gaps, such as homophony avoidance (Švedova 1982) are also unsatisfactory since they can, at best, explain only a small percentage of the gaps.

Thus, the data suggest that gaps persist in Russian primarily because they are not uttered, and this non-use is learned by succeeding generations of Russian speakers.[3] The clustering of the gaps among 2nd conjugation dental stems most likely is partially a remnant of their original causes, and partially represents analogic extension of gaps along morphophonological lines (see 2.3 below).

2.2 Empirical evidence for and operational definition of gaps

When dealing with descriptions in semi-prescriptive sources such as dictionaries, we must always ask whether they accurately represent language use. In other words, is there empirical evidence that speakers fail to use these words?

We sought evidence of gaps from the Russian National Corpus (RNC).[4] The RNC is a balanced textual corpus with 77.6 million words consisting primarily of the contemporary Russian literary language. The content is prose, plays, memoirs and biographies, literary criticism, newspaper and magazine articles, school texts, religious and

[1] Paradigmatic gaps also probably serve a sociolinguistic purpose, for example as markers of education, but sociolinguistic issues are beyond the scope of this paper.

[2] We use here the standard Cyrillic transliteration used by linguists. It should not be considered an accurate phonological representation. Elsewhere, when phonological issues are relevant, we use IPA.

[3] See Manning (2003) and Zuraw (2003) on learning from implicit negative evidence.

[4] Documentation: http://ruscorpora.ru/corpora-structure.html Mirror site used for searching: http://corpus.leeds.ac.uk/ruscorpora.html.

philosophical materials, technical and scientific texts, judicial and governmental publications, etc.

We gathered token frequencies for the six non-past forms of 3,265 randomly selected second conjugation verb lemmas. This produced 11,729 inflected forms with non-zero frequency.[5] As described in Section 3 below, these 11,729 form frequencies became our model's seed data.

To test the claim that Russian has verbal gaps, we examined a subsample of 557 2nd conjugation lemmas meeting the following criteria: (a) total non-past frequency greater than 36 raw tokens, and (b) 3sg and 3pl constituting less than 85% of total non-past frequency.[6] These constraints were designed to select verbs for which all six person-number combinations should be robustly attested, and to minimize sampling errors by removing lemmas with low attestation.

We calculated the probability of the 1sg inflection by dividing the number of 1sg forms by the total number of non-past forms. The subset was bimodally distributed with one peak near 0%, a trough at around 2%, and the other peak at 13.3%. The first peak represents lemmas in which the 1sg form is basically not used – gaps. Accordingly, we define gaps as second conjugation verbs which meet criteria (a) and (b) above, and for which the 1sg non-past form constitutes less than 2% of total non-past frequency for that lemma (N=56).

In accordance with the grammatical descriptions, our criteria are disproportionately likely to identify dental stems as gaps. Still, only 43 of 412 dental stems (10.4%) have gaps, compared with 13 gaps among 397 examples of other stems (3.3%).

Second, not all dental stems are equally affected. There seems to be a weak prototypicality effect centered around stems ending in /dʲ/, from which /tʲ/ and /zʲ/ each differ by one phonological feature. There may also be some weak semantic factors that we do not consider here.

/**dʲ**/	/**tʲ**/	/**zʲ**/	/**sʲ**/	/**stʲ**/
13.3% (19/143)	12.4% (14/118)	11.9% (5/42)	4.8% (3/62)	4.3% (2/47)

Table 1. Distribution of Russian verbal gaps among dental stems

[5] We excluded 29 high-frequency lemmas for which the corpus did not provide accurate counts.

[6] Russian has a number of verbs for which only the 3sg and 3pl are regularly used.

2.3 Some relevant historical facts

A significant difference between the morphological competition approach and our statistical learning approach is that the former attempts to provide a single account for both the rise and the perpetuation of paradigmatic gaps. By contrast, our statistical learning model does not require that the morphological system provide synchronic motivation. The following question thus arises: Were the Russian gaps originally caused by forces which are no longer in play in the language?

Baerman and Corbett (2006) find evidence that the gaps began with a single root, *-bed-* (e.g., *pobedit'* 'to win'), and subsequently spread analogically within dental stems. Baerman (2007) expands on the historical evidence, finding that a conspiracy of several factors provided the initial push towards defective 1sg forms. Most important among these, many of the verbs with 1sg gaps in modern Russian are historically associated with aberrant morphophonological alternations. He argues that when these unusual alternations were eliminated in the language, some of the words failed to be integrated into the new morphological patterns, which resulted in lexically specified gaps.

Important to the point here is that the elimination of marginal alternations removed an earlier synchronic motivation for the gaps. Yet gaps have persisted and new gaps have arisen (e.g., *pylesosit'* 'to vacuum'). This persistence is the behavior that we seek to model.

3 Formal aspects of the model

We take up two questions: How much machinery do we need for gaps to persist? How much machinery do we need for gaps to spread to phonologically similar words? We model three scenarios.

In the first scenario there is no grammar learning. Adult agents produce forms by random sampling from the forms that heard as children, and child agents hear those forms. In the subsequent generation children become adults. In this scenario there is thus no analogical pressure. Any perseverance of gaps results from word-specific learning.

The second scenario is similar to the first, except that the learning process includes analogical pressure from a random set of words. Specifically, for a target concept, the estimated distribution of its IPS is influenced by the distribution of known words. This enables the learner to express a known

concept with a novel IPS. For example, imagine that a learner hears the present tense verb form *googles*, but not the past tense *googled*. By analogy with other verbs, learners can expect the past tense to occur with a certain frequency, even if they have not encountered it.

The third scenario builds upon the second. In this version, the analogical pressure is not completely random. Instead, it is weighted by morphophonological similarity – similar word forms contribute more to the analogical force on a target concept than do dissimilar forms. This addition to the model is motivated by the pervasive importance of stem shape in the Russian morphological system generally, and potentially provides an account for the phonological prototypicality effect among Russian gaps.

The three scenarios thus represent increasing machinery for the model, and we use them to explore the conditions necessary for gaps to persist and spread. We created a multi-agent network model with Bayesian learning component. In the following sections we describe the model's structure, and outline the criteria by which we evaluate its output under the various conditions.

3.1 Social structure

Our model includes two generations of agents. Adult agents output linguistic forms, which provide linguistic input for child agents. Output/input occurs in batches.[7] After each batch all adults die, all children mature into adults, and a new generation of children is born. Each run of the model included 10 generations of agents.

We model the social structure with a random network. Each adult produces 100,000 verb forms, and each child is exposed to every production from every adult to whom they are connected. Each generation consisted of 50 adult agents, and child agents are connected to adults with some probability *p*. On average, each child agent is connected to 10 adult agents, meaning that each child hears, on average, 1,000,000 tokens.

3.2 Linguistic events

Russian gaps are localized to second conjugation non-past verb forms, so productions of these forms are the focus of interest. Formally, we define a linguistic event as a concept-inflection-form (C,I,F) triple. The concept serves to connect the different forms and inflections of the same lemma.

3.3 Definition of grammar

A grammar is defined as a probability distribution over linguistic events. This gives rise to natural formulations of learning and production as statistical processes: learning is estimating a probability distribution from existing data, and production is sampling from a probability distribution. The grammar can be factored into modular components:

$$p(C, I, F) = p(C) \cdot p(I \mid C) \cdot p(F \mid C, I)$$

In this paper we focus on the probability distribution of concept-inflection pairs. In other words, we focus on the relative frequency of inflectional property sets (IPS) on a lemma-by-lemma basis, represented by the middle term above.

Accordingly, we made the simplest possible assumptions for the first and last terms. To calculate the probability of a concept, children use the sample frequency (e.g., if they hear 10 tokens of the concept 'eat', and 1,000 tokens total, then p('eat') = 10/1000 = .01). Learning of forms is perfect. That is, learners always produce the correct form for every concept-inflection pair.

3.4 Learning model

Although production in the real world is governed by semantics, we treat it here as a statistical process, much like rolling a six-sided die which may or may not be fair. When producing a Russian non-past verb, there are six possible combinations of inflectional properties (3 persons * 2 numbers). In our model, word learning involves estimating the probability distribution over the frequencies of the six forms on a lemma-by-lemma basis. A hypothetical example that introduces our variables:

jest'	**1sg**	**2sg**	**3sg**	**1pl**	**2pl**	**3pl**	**SUM**
D	15	5	45	5	5	25	100
d	0.15	0.05	0.45	0.05	0.05	0.25	1

Table 2. Hypothetical probability distribution

The first row indicates the concept and the inflections. The second row (**D**) indicates the

[7] See Niyogi (2006) for why batch learning is a reasonable approximation in this context.

hypothetical number of tokens of *jest'* 'eat' that the learner heard for each inflection (bolding indicates a six-vector). We use |D| to indicate the sum of this row (=100), which is the concept frequency. The third row (**d**) indicates the sample probability of that inflection, which is simply the second row divided by |D|.

The learner's goal is to estimate the distribution that generated this data. We assume the multinomial distribution, whose parameter is simply the vector of probabilities of each IPS. For each concept, the learner's task is to estimate the probability of each IPS, represented by **h** in the equations below. We begin with Bayes' rule:

$$p(\mathbf{h} \mid \mathbf{D}) \propto p(\mathbf{h}) \cdot \text{multinom}(\mathbf{D} \mid \mathbf{h})$$

The prior distribution constitutes the analogical pressure on the lemma. It is generated from the "expected" behavior, $\mathbf{h^0}$, which is an average of the known behavior from a random sample of other lemmas. The parameter κ determines the number of lemmas that are sampled for this purpose – it represents how many existing words affect a new word. To model the effect of morphophonological similarity (mpSim), in one variant of the model we weight this average by the similarity of the stem-final consonant.[8] For example, this has the effect that existing dental stems have more of an effect on dental stems. In this case, we define

$$\mathbf{h^0} = \Sigma_{c' \text{ in sample}} \mathbf{d}_{c'} \cdot \text{mpSim}(c, c') / \Sigma\, \text{mpSim}(c, c')$$

We use a featural definition of similarity, so that if the stem-final consonants differ by 0, 1, 2, or 3 or more phonological features, the resulting similarity is 1, 2/3, 1/3, or 0, respectively.

The prior distribution should assign higher probability to hypotheses that are "closer" to this expected behavior $\mathbf{h^0}$. Since the hypothesis is itself a probability distribution, the natural measure to use is the KL divergence. We used an exponentially distributed prior with parameter β:

$$p(\mathbf{h}) \propto \exp(-\beta \cdot \mathbf{h^0} \,||\, \mathbf{h})$$

As will be shown shortly, β has a natural interpretation as the relative strength of the prior with respect to the observed data.

The learner calculates their final grammar by taking the mode of the posterior distribution (MAP). It can be shown that this value is given by

$$\arg\max p(\mathbf{h} \mid \mathbf{D}) = (\beta \cdot \mathbf{h^0} + |D| \cdot \mathbf{d}) / (\beta + |D|)$$

Thus, the output of this learning rule is a probability vector **h** that represents the estimated probability of each of the six possible IPS's for that concept. As can be seen from the equation above, this probability vector is an average of the expected behavior $\mathbf{h^0}$ and the observed data **d**, weighted by β and the amount of observed data |D|, respectively.

Our approach entails that from the perspective of a language learner, gaps are not qualitatively distinct from productive forms. Instead, 1sg non-past gaps represent one extreme of a range of probabilities that the first person singular will be produced. In this sense, "gaps" represent an artificial boundary which we place on a gradient structure for the purpose of evaluating our model.

The contrast between our learning model and the account of gaps presented in Albright (2003) merits emphasis at this point. Generally speaking, learning a word involves at least two tasks: learning how to generate the appropriate phonological form for a given concept and inflectional property set, and learning the probability that a concept and inflectional property set will be produced at all. Albright's model focuses on the former aspect; our model focuses on the latter. In short, our account of gaps lies in the likelihood of a concept-IPS pair being expressed, not in the likelihood of a form being expressed.

3.5 Production model

We model language production as sampling from the probability distribution that is the output of the learning rule.

3.6 Seeding the model

The input to the first generation was sampled from the verbs identified in the corpus search (see 2.2). Each input set contained 1,000,000 tokens, which was the average amount of input for agents in all succeeding generations. This made the first

[8] In Russian, the stem-final consonant is important for morphological behavior generally. Any successful Russian learner would have to extract the generalization, completely apart from the issues posed by gaps.

generation's input as similar as possible to the input of all succeeding generations.

3.7 Parameter space in the three scenarios

In our model we manipulate two parameters – the strength of the analogical force on a target concept during the learning process (β), and the number of concepts which create the analogical force (κ), taken randomly from known concepts.

As discussed above, we model three scenarios. In the first scenario, there is no grammar learning, so there is only one condition (β = 0). For the second and third scenarios, we run the model with four values for β, ranging from weak to strong analogical force (0.05, 0.25, 1.25, 6.25), and two values for κ, representing influence from a small or large set of other words (30, 300).

4 Evaluating the output of the model

We evaluate the output of our model against the following question: How well do gaps persist?

We count as gaps any forms meeting the criteria outlined in 2.2 above, tabulating the number of gaps which exist for only one generation, for two total generations, etc. We define τ as the expected number of generations (out of 10) that a given concept meets the gap criteria. Thus, τ represents a gap's "life expectancy" (see Figure 1).

We found that this distribution is exponential – there are few gaps that exist for all ten generations, and lots of gaps that exist for only one, so we calculated τ with a log linear regression. Each value reported is an average over 10 runs.

As discussed above, our goal was to discover whether the model can exhibit the same qualitative behavior as the historical development of Russian gaps. Persistence across a handful of generations (so far) and spread to a limited number of similar forms should be reflected by a non-negligible τ.

5 Results

In this section we present the results of our model under the scenarios and parameter settings above.

Remember that in the first scenario there is no grammar learning. This run of the model represents the baseline condition – completely word-specific knowledge. Sampling results in random walks on form frequencies, so once a word form disappears it never returns to the sample. Word-specific learning is thus sufficient for the perseverance of existing paradigmatic gaps and the creation of new ones. With no analogical pressure, gaps are robustly attested (τ = 6.32). However, the new gaps are not restricted to the 1sg, and under this scenario, learners are unable to generalize to a novel pairing of lexeme + IPS.

The second scenario presents a more complicated picture. As shown in Table 3, as analogical pressure (β) increases, gap life expectancy (τ) decreases. In other words, high analogical pressure quickly eliminates atypical frequency distributions, such as those exhibited by gaps. The runs with low values of β are particularly interesting because they represent an approximate balance between elimination of gaps as a general behavior, and the short-term persistence and even spread of gaps due to sampling artifacts and the influence of existing gaps. Thus, although the limit behavior is for gaps to disappear, this scenario retains the ability to explain persistence of gaps due to word-specific learning when there is weak analogical force.

At the same time, the facts of Russian differ from the behavior of the model in that the Russian gaps spread to morphophonologically similar forms, not random ones. The third version of our model weights the analogical strength of different concepts based upon morphophonological similarity to the target.

κ	**β**	**τ (random)**	**τ (phono.)**
--	0	6.32	
30	0.05	4.95	5.77
30	0.25	3.46	5.28
30	1.25	1.91	3.07
30	6.25	2.59	1.87
300	0.05	4.97	5.99
300	0.25	3.72	5.14
300	1.25	1.90	3.10
300	6.25	2.62	1.84

Table 3. Life expectancy of gaps, as a function of the strength of random analogical forces

Under these conditions we get two interesting results, presented in Table 3 above. First, gaps persist slightly better overall in scenario 3 than in

scenario 2 for all levels of κ and β.[9] Compare the τ values for random analogical force (scenario 2) with the τ values for morphophonologically weighted analogical force (scenario 3).

Second, strength of analogical force matters. When there is weak analogical pressure, weighting for morphophonological similarity has little effect on the persistence and spread of gaps. However, when there is relatively strong analogical pressure, morphophonological similarity helps atypical frequency distributions to persist, as shown in Figure 1. This results from the fact that there is a prototypicality effect for gaps. Since dental stems are more likely to be gaps, incorporating sensitivity to stem shape causes the analogical pressure on target dental stems to be relatively stronger from words that are gaps. Correspondingly, the analogical pressure on non-dental stems is relatively stronger from words that are not gaps. The prototypical stem shape for a gap is thereby perpetuated and gaps spread to new dental stems.

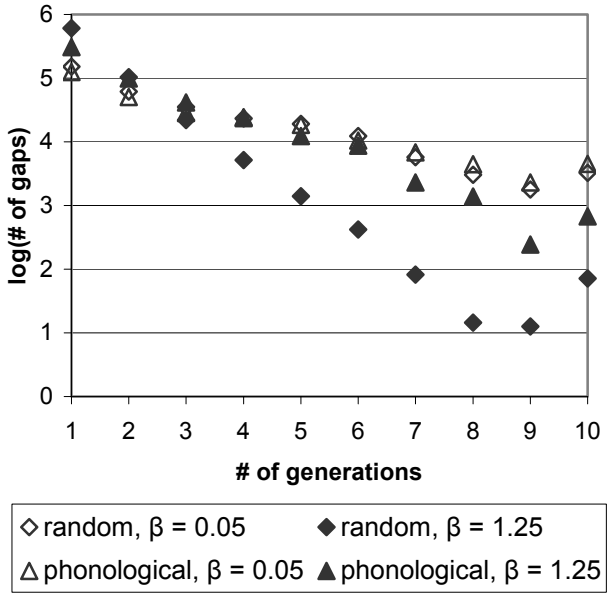


Figure 1. Gap life expectancy (β=0.05, κ=30)

[9] The apparent increase in gap half-life when β=6.25 is an artifact of the regression model. There were a few well-entrenched gaps whose high lemma frequency enables them to resist even high levels of analogical pressure over 10 generations. These data points skewed the regression, as shown by a much lower R^2 (0.5 vs. 0.85 or higher for all the other conditions).

6 Discussion

In conclusion, our model has in many respects succeeded in getting gaps to perpetuate and spread. With word-specific learning alone, well-entrenched gaps can be maintained across multiple generations. More significantly, weak analogical pressure, especially if weighted for morpho-phonological similarity, results in the perseverance and short-term growth of gaps. This is essentially the historical pattern of the Russian verbal gaps.

These results highlight several issues regarding both the nature of paradigmatic gaps and the structure of inflectional systems generally.

We claim that it is not necessary to posit an irreconcilable conflict in the generation of inflected forms in order to account for gaps. Remember that in our model, agents face no conflict in terms of which form to produce – there is only one possibility. Yet the gaps persist in part because of analogical pressure from existing gaps. Albright (2003) himself is agnostic on the issue of whether form-based competition is *necessary* for the existence and persistence of gaps, but Hudson (2000), among others, claims that gaps could not exist in the absence of it. We have presented evidence that this claim is unfounded.

But why would someone assume that grammar competition is necessary? Hudson's claim arises from a confusion of two issues. Discussing the English paradigmatic gap *amn't*, Hudson states that "a simple application of [the usage-based learning] principle would be to say that the gap exists simply because nobody says *amn't*... But this explanation is too simple... There are many inflected words that may never have been uttered, but which we can nevertheless imagine ourselves using, given the need; we generate them by generalization" (Hudson 2000:300). By his logic, there must therefore be some source of grammar conflict which prevents speakers from generalizing.

However, there is a substantial difference between having no information about a word, and having information about the non-usage of a word. We do not dispute learners' ability to generalize. We only claim that information of non-usage is sufficient to block such generalizations. When confronted with a new word, speakers will happily generalize a word form, but this is not the same task that they perform when faced with gaps.

The perseverance of gaps in the absence of form-based competition shows that a different, non-form level of representation is at issue. Generating inflectional morphology involves at least two different types of knowledge: knowledge about the appropriate word form to express a given concept and IPS on the one hand, and knowledge of how often that concept and IPS is expressed on the other. The emergence of paradigmatic gaps may be closely tied to the first type of knowledge, but the Russian gaps, at least, persist because of the second type of knowledge. We therefore propose that morphology may be defective at the morphosyntactic level.

This returns us to the question that we began this paper with – how paradigmatic gaps can persist in light of the overwhelming productivity of inflectional morphology. Our model suggests that the apparent contradiction is, at least in some cases, illusory. Productivity refers to the likelihood of a given inflectional pattern applying to a given combination of stem and IPS. Our account is based in the likelihood of the stem and inflectional property set being expressed at all, regardless of the form. In short, the Russian paradigmatic gaps represent an issue which is orthogonal to productivity. The two issues are easily confused, however. An unusual frequency distribution can make it *appear* that there is in fact a problem at the level of form, even when there may not be.

Finally, our simulations raise the question of whether the 1sg non-past gaps in Russian will persist in the language in the long term. In our model, analogical forces delay convergence to the mean, but the limit behavior is that all gaps disappear. Although there is evidence in Russian that words can develop new gaps, we do not know with any great accuracy whether the set of gaps is currently expanding, contracting, or approximately stable. Our model predicts that in the long run, the gaps will disappear under general analogical pressure. However, another possibility is that our model includes only enough factors (e.g., morphophonological similarity) to approximate the short-term influences on the Russian gaps and that we would need more factors, such as semantics, to successfully model their long-term development. This remains an open question.

References

Albright, Adam. 2003. A quantitative study of Spanish paradigm gaps. In *West Coast Conference on Formal Linguistics 22 proceedings*, eds. Gina Garding and Mimu Tsujimura. Somerville, MA: Cascadilla Press, 1-14.

Albright, Adam, and Bruce Hayes. 2002. Modeling English past tense intuitions with minimal generalization. In *Proceedings of the Sixth Meeting of the Association for Computational Linguistics Special Interest Group in Computational Phonology in Philadelphia, July 2002*, ed. Michael Maxwell. Cambridge, MA: Association for Computational Linguistics, 58-69.

Baerman, Matthew. 2007. The diachrony of defectiveness. Paper presented at 43rd Annual Meeting of the Chicago Linguistic Society in Chicago, IL, May 3-5, 2007.

Baerman, Matthew, and Greville Corbett. 2006. Three types of defective paradigms. Paper presented at The Annual Meeting of the Linguistic Society of America in Albuquerque, NM, January 5-8, 2006.

Hudson, Richard. 2000. *I amn't. *Language* 76 (2):297-323.

Manning, Christopher. 2003. Probabilistic syntax. In *Probabilistic linguistics*, eds. Rens Bod, Jennifer Hay and Stephanie Jannedy. Cambridge, MA: MIT Press, 289-341.

Niyogi, Partha. 2006. *The computational nature of language learning and evolution*. Cambridge, MA: MIT Press.

Sims, Andrea. 2006. *Minding the gaps: Inflectional defectiveness in paradigmatic morphology*. Ph.D. thesis: Linguistics Department, The Ohio State University.

Švedova, Julja. 1982. *Grammatika sovremennogo russkogo literaturnogo jayzka*. Moscow: Nauka.

Zaliznjak, A.A., ed. 1977. *Grammatičeskij slovar' russkogo jazyka: Slovoizmenenie*. Moskva: Russkij jazyk.

Zuraw, Kie. 2003. Probability in language change. In *Probabilistic linguistics*, eds. Rens Bod, Jennifer Hay and Stephanie Jannedy. Cambridge, MA: MIT Press, 139-176.

Substring-Based Transliteration

Tarek Sherif and **Grzegorz Kondrak**
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2E8
{tarek,kondrak}@cs.ualberta.ca

Abstract

Transliteration is the task of converting a word from one alphabetic script to another. We present a novel, substring-based approach to transliteration, inspired by phrase-based models of machine translation. We investigate two implementations of substring-based transliteration: a dynamic programming algorithm, and a finite-state transducer. We show that our substring-based transducer not only outperforms a state-of-the-art letter-based approach by a significant margin, but is also orders of magnitude faster.

1 Introduction

A significant proportion of out-of-vocabulary words in machine translation models or cross language information retrieval systems are named entities. If the languages are written in different scripts, these names must be transliterated. Transliteration is the task of converting a word from one writing script to another, usually based on the phonetics of the original word. If the target language contains all the phonemes used in the source language, the transliteration is straightforward. For example, the Arabic transliteration of *Amanda* is أماندا, which is essentially pronounced in the same way. However, if some of the sounds are missing in the target language, they are generally mapped to the most phonetically similar letter. For example, the sound [p] in the name *Paul*, does not exist in Arabic, and the phonotactic constraints of Arabic disallow the sound [ɑ] in this context, so the word is transliterated as بول, pronounced [bul].

The information loss inherent in the process of transliteration makes back-transliteration, which is the restoration of a previously transliterated word, a particularly difficult task. Any phonetically reasonable forward transliteration is essentially correct, although occasionally there is a standard transliteration (e.g. *Omar Sharif*). In the original script, however, there is usually only a single correct form. For example, both *Naguib Mahfouz* and *Najib Mahfuz* are reasonable transliterations of نجيب محفوظ, but *Tsharlz Dykens* is certainly not acceptable if one is referring to the author of *Oliver Twist*.

In a statistical approach to machine transliteration, given a foreign word F, we are interested in finding the English word $\hat{E}$ that maximizes $P(E|F)$. Using Bayes' rule, and keeping in mind that F is constant, we can formulate the task as follows:

$$\hat{E} = \arg\max_E \frac{P(F|E)P(E)}{P(F)}$$

$$= \arg\max_E P(F|E)P(E)$$

This is known as the noisy channel approach to machine transliteration, which splits the task into two parts. The language model provides an estimate of the probability $P(E)$ of an English word, while the transliteration model provides an estimate of the probability $P(F|E)$ of a foreign word being a transliteration of an English word. The probabilities assigned by the transliteration and language models counterbalance each other. For example, simply concatenating the most common mapping for each letter in the Arabic string مايكل, produces the string *maykl*, which is barely pronounceable. In order to generate the correct $Michael$, a model needs

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 944–951,
Prague, Czech Republic, June 2007.

to know the relatively rare letter relationships *ch*/ﭺ and *ae*/ϵ, and to balance their unlikelihood against the probability of the correct transliteration being an actual English name.

The search for the optimal English transliteration $\hat{E}$ for a given foreign name F is referred to as decoding. An efficient approach to decoding is dynamic programming, in which solutions to subproblems are maintained in a table and used to build up the global solution in a bottom-up approach. Dynamic programming approaches are optimal as long as the dynamic programming invariant assumption holds. This assumption states that if the optimal path through a graph happens to go through state *q*, then this optimal path must include the best path up to and including *q*. Thus, once an optimal path to state *q* is found, all other paths to *q* can be eliminated from the search. The validity of this assumption depends on the state space used to define the model. Typically, for problems related to word comparison, a dynamic programming approach will define states as positions in the source and target words. As will be shown later, however, not all models can be represented with such a state space.

The phrase-based approach developed for statistical machine translation (Koehn et al., 2003) is designed to overcome the restrictions on many-to-many mappings in word-based translation models. This approach is based on learning correspondences between phrases, rather than words. Phrases are generated on the basis of a word-to-word alignment, with the constraint that no words within the phrase pair are linked to words outside the phrase pair.

In this paper, we propose to apply phrase-based translation methods to the task of machine transliteration, in an approach we refer to as substring-based transliteration. We consider two implementations of these models. The first is an adaptation of the monotone search algorithm outlined in (Zens and Ney, 2004).The second encodes the substring-based transliteration model as a transducer. The results of experiments on Arabic-to-English transliteration show that the substring-based transducer outperforms a state-of-the-art letter-based transducer, while at the same time being orders of magnitude smaller and faster.

The remainder of the paper is organized as follows. Section 2 discusses previous approaches to machine transliteration. Section 3 presents the letter-based transducer approach to Arabic-English transliteration proposed in (Al-Onaizan and Knight, 2002), which we use as the main point of comparison for our substring-based models. Section 4 presents our substring-based approaches to transliteration. In Section 5, we outline the experiments used to evaluate the models and present their results. Finally, Section 6 contains our overall impressions and conclusions.

2 Previous Work

Arababi et al. (1994) propose to model forward transliteration through a combination of neural net and expert systems. Their main task was to vowelize the Arabic names as a preprocessing step for transliteration. Their method is Arabic-specific and requires that the Arabic names have a regular pattern of vowelization.

Knight and Graehl (1998) model the transliteration of Japanese syllabic *katakana* script into English with a sequence of finite-state transducers. After performing a conversion of the English and katakana sequences to their phonetic representations, the correspondences between the English and Japanese phonemes are learned with the expectation maximization (EM) algorithm. Stalls and Knight (1998) adapt this approach to Arabic, with the modification that the English phonemes are mapped directly to Arabic letters. Al-Onaizan and Knight (2002) find that a model mapping directly from English to Arabic letters outperforms the phoneme-to-letter model.

AbdulJaleel and Larkey (2003) model forward transliteration from Arabic to English by treating the words as sentences and using a statistical word alignment model to align the letters. They select common English n-grams based on cases when the alignment links an Arabic letter to several English letters, and consider these n-grams as single letters for the purpose of training. The English transliterations are produced using probabilities, learned from the training data, for the mappings between Arabic letters and English letters/n-grams.

Li et al. (2004) propose a letter-to-letter n-gram transliteration model for Chinese-English transliteration in an attempt to allow for the encoding of more

contextual information. The model isolates individual mapping operations between training pairs, and then learns n-gram probabilities for sequences of these mapping operations. Ekbal et al. (2006) adapt this model to the transliteration of names from Bengali to English.

3 Letter-based Transliteration

The main point of comparison for the evaluation of our substring-based models of transliteration is the letter-based transducer proposed by (Al-Onaizan and Knight, 2002). Their model is a composition of a transliteration transducer and a language transducer. Mappings in the transliteration transducer are defined between 1-3 English letters and 0-2 Arabic letters, and their probabilities are learned by EM. The transliteration transducer is split into three states to allow mapping probabilities to be learned separately for letters at the beginning, middle and end of a word. Unlike the transducers proposed in (Stalls and Knight, 1998) and (Knight and Graehl, 1998) no attempt is made to model the pronunciation of words. Although names are generally transliterated based on how they sound, not how they look, the letter-phoneme conversion itself is problematic as it is not a trivial task. Many transliterated words are proper names, whose pronunciation rules may vary depending on the language of origin (Li et al., 2004). For example, ch is generally pronounced as either [ʧ] or [k] in English names, but as [ʃ] in French names.

The language model is implemented as a finite state acceptor using a combination of word unigram and letter trigram probabilities. Essentially, the word unigram model acts as a probabilistic lookup table, allowing for words seen in the training data to be produced with high accuracy, while the letter trigram probabilities are used model words not seen in the training data.

4 Substring-based Transliteration

Our substring-based transliteration approach is an adaptation of phrase-based models of machine translation to the domain of transliteration. In particular, our methods are inspired by the monotone search algorithm proposed in (Zens and Ney, 2004). We introduce two models of substring-based transliteration: the Viterbi substring decoder and the substring-based transducer. Table 1 presents a comparison of the substring-based models to the letter-based model discussed in Section 3.

4.1 The Monotone Search Algorithm

Zens and Ney (2004) propose a linear-time decoding algorithm for phrase-based machine translation. The algorithm requires that the translation of phrases be sequential, disallowing any phrase reordering in the translation.

Starting from a word-based alignment for each pair of sentences, the training for the algorithm accepts all contiguous bilingual phrase pairs (up to a predetermined maximum length) whose words are only aligned with each other (Koehn et al., 2003). The probabilities $P(\tilde{f}|\tilde{e})$ for each foreign phrase $\tilde{f}$ and English phrase $\tilde{e}$ are calculated on the basis of counts gleaned from a bitext. Since the counting process is much simpler than trying to learn the phrases with EM, the maximum phrase length can be made arbitrarily long with minimal jumps in complexity. This allows the model to actually encode contextual information into the translation model instead of leaving it completely to the language model. There are no null (ϵ) phrases so the model does not handle insertions or deletions explicitly. They can be handled implicitly, however, by including inserted or deleted words as members of a larger phrase.

Decoding in the monotone search algorithm is performed with a Viterbi dynamic programming approach. For a foreign sentence of length J and a phrase length maximum of M, a table is filled with a row j for each position in the input foreign sentence, representing a translation sequence ending at that foreign word, and each column e represents possible final English words for that translation sequence. Each entry in the table Q is filled according to the following recursion:

$$\begin{aligned} Q(0,\$) &= 1 \\ Q(j,e) &= \max_{e',\tilde{e},\tilde{f}} P(\tilde{f}|\tilde{e})P(\tilde{e}|e')Q(j',e') \\ Q(J+1,\$) &= \max_{e'} Q(J,e')P(\$|e') \end{aligned}$$

where $\tilde{f}$ is a foreign phrase beginning at $j'+1$, ending at j and consisting of up to M words. The '$' symbol is the sentence boundary marker.

	Letter Transducer	Viterbi Substring	Substring Transducer
Model Type	Transducer	Dynamic Programming	Transducer
Transliteration Model	Letter	Substring	Substring
Language Model	Word/Letter	Substring/Letter	Word/Letter
Null Symbols	Yes	No	No
Alignments	All	Most Probable	Most Probable

Table 1: Comparison of statistical transliteration models.

In the above recursion, the language model is represented as $P(\tilde{e}|e')$, the probability of the English phrase given the previous English word. Because of data sparseness issues in the context of word phrases, the actual implementation approximates this probability using word n-grams.

4.2 Viterbi Substring Decoder

We propose to adapt the monotone search algorithm to the domain of transliteration by substituting letters and substrings for the words and phrases of the original model. There are, in fact, strong indications that the monotone search algorithm is better suited to transliteration than it is to translation. Unlike machine translation, where the constraint on reordering required by monotone search is frequently violated, transliteration is an inherently sequential process. Also, the sparsity issue in training the language model is much less pronounced, allowing us to model $P(\tilde{e}|e')$ directly.

In order to train the model, we extract the one-to-one Viterbi alignment of a training pair from a stochastic transducer based on the model outlined in (Ristad and Yianilos, 1998). Substrings are then generated by iteratively appending adjacent links or unlinked letters to the one-to-one links of the alignment. For example, assuming a maximum substring length of 2, the <*r*, ر> link in the alignment presented in Figure 1 would participate in the following substring pairs: <*r*, ر>, <*ur*, ر>, and <*ra*, را>.

The fact that the Viterbi substring decoder employs a dynamic programming search through the source/target letter state space described in Section 1 renders the use of a word unigram language model impossible. This is due to the fact that alternate paths to a given source/target letter pair are being eliminated as the search proceeds. For example, suppose the Viterbi substring decoder were given the

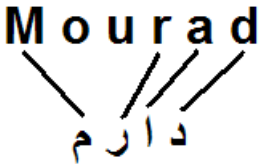


Figure 1: A one-to-one alignment of *Mourad* and مراد. For clarity the Arabic name is written left to right.

Arabic string كريم, and there are two valid English names in the language model, *Karim* (the correct transliteration of the input) and *Kristine* (the Arabic transliteration of which would be كرستين). The optimal path up to the second letter might go through <ك,*k*>, <ر,*r*>. At this point, it is transliterating into the name *Kristine*, but as soon as it hits the third letter (ي), it is clear that this is the incorrect choice. In order to recover from the error, the search would have to backtrack to the beginning and return to state <ر,*r*> from a different path, but this is an impossibility since all other paths to that state have been eliminated from the search.

4.3 Substring-based Transducer

The major advantage the letter-based transducer presented in Section 3 has over the Viterbi substring decoder is its word unigram language model, which allows it to reproduce words seen in the training data with high accuracy. On the other hand, the Viterbi substring decoder is able to encode contextual information in the transliteration model because of its ability to consider larger many-to-many mappings. In a novel approach presented here, we propose a substring-based transducer that draws on both advantages. The substring transliteration model learned for the Viterbi substring decoder is encoded as a transducer, thus allowing it to use a word uni-

gram language model. Our model, which we refer to as the substring-based transducer, has several advantages over the previously presented models.

- The substring-based transducer can be composed with a word unigram language model, allowing it to transliterate names seen in training for the language model with greater accuracy.
- Longer many-to-many mappings enable the transducer to encode contextual information into the transliteration model. Compared to the letter-based transducer, it allows for the generation of longer well-formed substrings (or potentially even entire words).
- The letter-based transducer considers all possible alignments of the training examples, meaning that many low-probability mappings are encoded into the model. This issue is even more pronounced in cases where the desired transliteration is not in the word unigram model, and it is guided by the weaker letter trigram model. The substring-based transducer can eliminate many of these low-probability mappings because of its commitment to a single high-probability one-to-one alignment during training.
- A major computational advantage this model has over the letter-based transducer is the fact that null characters (ϵ) are not encoded explicitly. Since the Arabic input to the letter-based transducer could contain an arbitrary number of nulls, the potential number of output strings from the transliteration transducer is infinite. Thus, the composition with the language transducer must be done in such a way that there is a valid path for all of the strings output by the transliteration transducer that have a positive probability in the language model. This leads to prohibitively large transducers. On the other hand, the substring-based transducer handles nulls implicitly (e.g. the mapping *ke*:ك implicitly represents *e*:ϵ after a *k*), so the transducer itself is not required to deal with them.

5 Experiments

In this section, we describe the evaluation of our models on the task of Arabic-to-English transliteration.

5.1 Data

For our experiments, we required bilingual name pairs for testing and development data, as well as for the training of the transliteration models. To train the language models, we simply needed a list of English names. Bilingual data was extracted from the Arabic-English Parallel News part 1 (approx. 2.5M words) and the Arabic Treebank Part 1-10k word English Translation. Both bitexts contain Arabic news articles and their English translations. The English name list for the language model training was extracted from the English-Arabic Treebank v1.0 (approx. 52k words)[1]. The language model training set consisted of all words labeled as proper names in this corpus along with all the English names in the transliteration training set. Any names in any of the data sets that consisted of multiple words (e.g. first name/last name pairs) were split and considered individually. Training data for the transliteration model consisted of 2844 English-Arabic pairs. The language model was trained on a separate set of 10991 (4494 unique) English names. The final test set of 300 English-Arabic transliteration pairs contained no overlap with the set that was used to induce the transliteration models.

5.2 Evaluation Methodology

For each of the 300 transliteration pairs in the test set, the name written in Arabic served as input to the models, while its English counterpart was considered a gold standard transliteration for the purpose of evaluation. Two separate tests were performed on the test set. In the first, the 300 English words in the test set were added to the training data for the language models (the *seen* test), while in the second, all English words in the test set were removed from the language model's training data (the *unseen* test). Both tests were run on the same set of words to ensure that variations in performance for *seen* and *unseen* words were solely due to whether or not they appear in the language model (and not, for example, their language of origin). The *seen* test is similar to tests run in (Knight and Graehl, 1998) and (Stalls and Knight, 1998) where the models could not produce any words not included in the language

[1] All corpora are distributed by the Linguistic Data Consortium. Despite the name, the English-Arabic Treebank v1.0 contains only English data.

model training data. The models were evaluated on the *seen* test set in terms of exact matches to the gold standard. Because the task of generating transliterations for the *unseen* test set is much more difficult, exact match accuracy will not provide a meaningful metric for comparison. Thus, a softer measure of performance was required to indicate how close the generated transliterations are to the gold standard. We used Levenshtein distance: the number of insertions, deletions and substitutions required to convert one string into another. We present the results separately for names of Arabic origin and for those of non-Arabic origin.

We also performed a third test on words that appear in both the transliteration and language model training data. This test was not indicative of the overall strength of the models but was meant to give a sense of how much each model depends on its language model versus its transliteration model.

5.3 Setup

Five approaches were evaluated on the Arabic-English transliteration task.

- **Baseline:** As a baseline for our experiments, we used a simple deterministic mapping algorithm which maps Arabic letters to the most likely letter or sequence of letters in English.
- **Letter-based Transducer:** Mapping probabilities were learned by running the forward-backward algorithm until convergence. The language model is a combination of word unigram and letter trigram models and selects a word unigram or letter trigram modeling of the English word depending on whichever one assigns the highest probability. The letter-based transducer was implemented in Carmel[2].
- **Viterbi Substring Decoder:** We experimented with maximum substring lengths between 3 and 10 on the development set, and found that a maximum length of 6 was optimal.
- **Substring-based Transducer:** The substring-based transducer was also implemented in Carmel. We found that this model worked best with a maximum substring length of 4.

[2]Carmel is a finite-state transducer package written by Jonathan Graehl. It is available at http://www.isi.edu/licensed-sw/carmel/.

Method	Arabic	Non-Arabic	All
Baseline	1.9	2.1	2.0
Letter trans.	45.9	64.3	54.7
Viterbi substring	15.9	30.1	22.7
Substring trans.	**59.9**	**81.1**	**70.0**
Human	33.1	40.6	36.7

Table 2: Exact match accuracy percentage on the *seen* test set for various methods.

Method	Arabic	Non-Arabic	All
Baseline	2.32	2.80	2.55
Letter trans.	2.46	2.63	2.54
Viterbi substring	**1.90**	**2.13**	**2.01**
Substring trans.	1.92	2.41	2.16
Human	1.24	1.42	1.33

Table 3: Average Levenshtein distance on the *unseen* test set for various methods.

- **Human:** For the purpose of comparison, we allowed an independent human subject (fluent in Arabic, but a native speaker of English) to perform the same task. The subject was asked to transliterate the Arabic words in the test set without any additional context. No additional resources or collaboration were allowed.

5.4 Results on the Test Set

Table 2 presents the word accuracy performance of each transliterator when the test set is available to the language models. Table 3 shows the average Levenshtein distance results when the test set is unavailable to the language models. Exact match performance by the automated approaches on the *unseen* set did not exceed 10.3% (achieved by the Viterbi substring decoder). Results on the *seen* test suggest that non-Arabic words (back transliterations) are easier to transliterate exactly, while results for the *unseen* test suggest that errors on Arabic words (forward transliterations) tend to be closer to the gold standard.

Overall, our substring-based transducer clearly outperforms the letter-based transducer. Its performance is better in both tests, but its advantage is particularly pronounced on words it has seen in the training data for the language model (the task

	Arabic	LBT	SBT	Correct
1	عثمان	Uthman	Uthman	Othman
2	اشرف	Asharf	Asharf	Ashraf
3	رفعت	Rafeet	Arafat	Refaat
4	اسامة	Istamaday	Asuma	Usama
5	ايمان	Erdman	Aliman	Iman
6	ووتش	Wortch	Watch	Watch
7	ميلز	Mellis	Mills	Mills
8	فيراري	February	Firari	Ferrari

Table 4: A sample of the errors made by the letter-based (LBT) and segment-based (SBT) transducers.

for which the letter-based transducer was originally designed). Since both transducers use exactly the same language model, the fact that the substring-based transducer outperforms the letter-based transducer indicates that it learns a stronger transliteration model.

The Viterbi substring decoder seems to struggle when it comes to recreating words seen the language training data, as evidenced by its weak performance on the *seen* test. Obviously, its substring/letter bigram language model is no match for the word unigram model used by the transducers on this task. On the other hand, its stronger performance on the *unseen* test set suggests that its language model is stronger than the letter trigram used by the transducers when it comes to generating completely novel words.

A sample of the errors made by the letter- and substring-based transducers is presented in Table 4. In general, when both models err, the substring-based transducer tends toward more phonetically reasonable choices. The most common type of error is simply correct alternate English spellings of an Arabic name (error 1). Error 2 is an example of a learned mapping being misplaced (the deleted *a*). Error 3 indicates that the letter-based transducer is able to avoid these misplaced mappings at the beginning or end of a word because of its three-state transliteration transducer (i.e. it learns not to allow vowel deletions at the beginning of a word). Errors 4 and 5 are cases where the letter-based transducer produced particularly awkward transliterations. Errors 6 and 7 are names that actually appear in the word unigram model but were missed by the letter-based transducer, while error 8 is an example of the letter-based transducer incorrectly choosing a name from the word unigram model. As discussed in Section 4.3, this is likely due to mappings learned from low-probability alignments.

Method	Exact match	Avg Lev.
Letter transducer	81.2	0.46
Viterbi substring	83.2	0.24
Substring transducer	**94.4**	**0.09**

Table 5: Results for testing on the transliteration training set.

5.5 Results on the Training Set

The substring-based approaches encode a great deal of contextual information into the transliteration model. In order to assess how much the performance of each approach depends on its language model versus its transliteration model, we tested the three statistical models on the set of 2844 names seen in both the transliteration and language model training. The results of this experiment are presented in Table 5. The Viterbi substring decoder receives the biggest boost, outperforming the letter-based transducer, which indicates that its strength lies mainly in its transliteration modeling as opposed to its language modeling. The substring-based transducer, however, still outperforms it by a large margin, achieving near-perfect results. Most of the remaining errors can be attributed to names with alternate correct spellings in English.

The results also suggest that the substring-based transducer practically subsumes a naive "lookup table" approach. Although the accuracy achieved is less than 100%, the substring-based transducer has the great advantage of being able to handle noise in the input. In other words, if the spelling of an input word does not match an Arabic word from the training data, a lookup table will generate nothing, while the substring-based transducer could still search for the correct transliteration.

5.6 Computational Considerations

Another point of comparison between the models is complexity. The letter-based transducer encodes 56144 mappings while the substring-based transducer encodes 13948, but as shown in Table 6, once

Method	Size (states/arcs)
Letter transducer	86309/547184
Substring transducer	759/2131

Table 6: Transducer sizes for composition with the word حلمي (*Helmy*).

Method	Time
Letter transducer	5h52min
Viterbi substring	3 sec
Substring transducer	11 sec

Table 7: Running times for the 300 word test set.

the transducers are fully composed, the difference becomes even more pronounced. As discussed in Section 4.3, the reason for the size explosion factor in the letter-based transducer is the possibility of null characters in the input word.

The running times for the statistical approaches on the 300 word test set are presented in Table 7. The huge computational advantage of the substring-based approach makes it a much more attractive option for any real-world application. Tests were performed on an AMD Athlon 64 3500+ machine with 2GB of memory running Red Hat Enterprise Linux release 4.

6 Conclusion

In this paper, we presented a new substring-based approach to modeling transliteration inspired by phrase-based models of machine translation. We tested both dynamic programming and finite-state transducer implementations, the latter of which enabled us to use a word unigram language model to improve the accuracy of generated transliterations. The results of evaluation on the task of Arabic-English transliteration indicate that the substring-based approach not only improves performance over a state-of-the-art letter-based model, but also leads to major gains in efficiency. Since no language-specific information was encoded directly into the models, they can also be used for transliteration between other language pairs.

In the future, we plan to consider more complex language models in order to improve the results on unseen words, which should certainly be feasible for the substring-based transducer because of its efficient memory usage. Another feature of the substring-based transducer that we have not yet explored is its ability to easily produce an n-best list of transliterations. We plan to investigate whether using methods like discriminative reranking (Och and Ney, 2002) on such an n-best list could improve performance.

Acknowledgments

We would like to thank Colin Cherry and the other members of the NLP research group at the University of Alberta for their helpful comments. This research was supported by the Natural Sciences and Engineering Research Council of Canada.

References

N. AbdulJaleel and L. S. Larkey. 2003. Statistical transliteration for English-Arabic cross language information retrieval. In *CIKM*, pages 139–146.

Y. Al-Onaizan and K. Knight. 2002. Machine transliteration of names in Arabic text. In *ACL Workshop on Comp. Approaches to Semitic Languages.*

M. Arababi, S.M. Fischthal, V.C. Cheng, and E. Bart. 1994. Algorithmns for Arabic name transliteration. *IBM Journal of Research and Development*, 38(2).

A. Ekbal, S.K. Naskar, and S. Bandyopadhyay. 2006. A modified joint source-channel model for transliteration. In *COLING/ACL Poster Sessions*, pages 191–198.

K. Knight and J. Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599–612.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *NAACL-HLT*, pages 48–54.

H. Li, M. Zhang, and J. Su. 2004. A joint source-channel model for machine transliteration. In *ACL*, pages 159–166.

F. J. Och and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *ACL*, pages 295–302.

E. S. Ristad and P. N. Yianilos. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532.

B. Stalls and K. Knight. 1998. Translating names and technical terms in Arabic text. In *COLING/ACL Workshop on Comp. Approaches to Semitic Languages*.

R. Zens and H. Ney. 2004. Improvements in phrase-based statistical machine translation. In *HLT-NAACL*, pages 257–264.

Pipeline Iteration

Kristy Hollingshead and **Brian Roark**
Center for Spoken Language Understanding, OGI School of Science & Engineering
Oregon Health & Science University, Beaverton, Oregon, 97006 USA
{hollingk,roark}@cslu.ogi.edu

Abstract

This paper presents pipeline iteration, an approach that uses output from later stages of a pipeline to constrain earlier stages of the same pipeline. We demonstrate significant improvements in a state-of-the-art PCFG parsing pipeline using base-phrase constraints, derived either from later stages of the parsing pipeline or from a finite-state shallow parser. The best performance is achieved by reranking the union of unconstrained parses and relatively heavily-constrained parses.

1 Introduction

A "pipeline" system consists of a sequence of processing stages such that the output from one stage provides the input to the next. Each stage in such a pipeline identifies a subset of the possible solutions, and later stages are constrained to find solutions within that subset. For example, a part-of-speech tagger could constrain a "base phrase" chunker (Ratnaparkhi, 1999), or the n-best output of a parser could constrain a reranker (Charniak and Johnson, 2005). A pipeline is typically used to reduce search complexity for rich models used in later stages, usually at the risk that the best solutions may be pruned in early stages.

Pipeline systems are ubiquitous in natural language processing, used not only in parsing (Ratnaparkhi, 1999; Charniak, 2000), but also machine translation (Och and Ney, 2003) and speech recognition (Fiscus, 1997; Goel et al., 2000), among others. Despite the widespread use of pipelines, they have been understudied, with very little work on general techniques for designing and improving pipeline systems (although *cf.* Finkel et al. (2006)). This paper presents one such general technique, here applied to stochastic parsing, whereby output from later stages of a pipeline is used to constrain earlier stages of the same pipeline. To our knowledge, this is the first time such a pipeline architecture has been proposed.

It may seem surprising that later stages of a pipeline, already constrained to be consistent with the output of earlier stages, can profitably inform the earlier stages in a second pass. However, the richer models used in later stages of a pipeline provide a better distribution over the subset of possible solutions produced by the early stages, effectively resolving some of the ambiguities that account for much of the original variation. If an earlier stage is then constrained in a second pass not to vary with respect to these resolved ambiguities, it will be forced to find other variations, which may include better solutions than were originally provided.

To give a rough illustration, consider the Venn diagram in Fig. 1(i). Set A represents the original subset of possible solutions passed along by the earlier stage, and the dark shaded region represents high-probability solutions according to later stages. If some constraints are then extracted from these high-probability solutions, defining a subset of solutions (S) that rule out some of A, the early stage will be forced to produce a different set (B). Constraints derived from later stages of the pipeline focus the search in an area believed to contain high-quality candidates.

Another scenario is to use a different model altogether to constrain the pipeline. In this scenario,

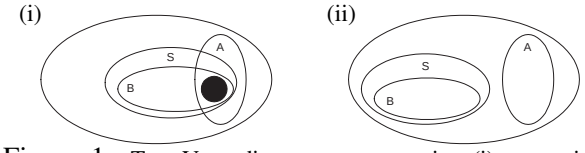


Figure 1: Two Venn diagrams, representing (i) constraints derived from later stages of an iterated pipelined system; and (ii) constraints derived from a different model.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 952–959,
Prague, Czech Republic, June 2007.

represented in Fig. 1(ii), the other model constrains the early stage to be consistent with some subset of solutions (S), which may be largely or completely disjoint from the original set A. Again, a different set (B) results, which may include better results than A. Whereas when iterating we are guaranteed that the new subset S will overlap at least partially with the original subset A, that is not the case when making use of constraints from a separately trained model.

In this paper, we investigate pipeline iteration within the context of the Charniak and Johnson (2005) parsing pipeline, by constraining parses to be consistent with a base-phrase tree. We derive these base-phrase constraints from three sources: the reranking stage of the parsing pipeline; a finite-state shallow parser (Hollingshead et al., 2005); and a combination of the output from these two sources. We compare the relative performance of these three sources and find the best performance improvements using constraints derived from a weighted combination of shallow parser output and reranker output.

The Charniak parsing pipeline has been extensively studied over the past decade, with a number of papers focused on improving early stages of the pipeline (Charniak et al., 1998; Caraballo and Charniak, 1998; Blaheta and Charniak, 1999; Hall and Johnson, 2004; Charniak et al., 2006) as well as many focused on optimizing final parse accuracy (Charniak, 2000; Charniak and Johnson, 2005; McClosky et al., 2006). This focus on optimization has made system improvements very difficult to achieve; yet our relatively simple architecture yields statistically significant improvements, making pipeline iteration a promising approach for other tasks.

2 Approach

Our approach uses the Charniak state-of-the-art parsing pipeline. The well-known Charniak (2000) coarse-to-fine parser is a two-stage parsing pipeline, in which the first stage uses a vanilla PCFG to populate a chart of parse constituents. The second stage, constrained to only those items in the first-stage chart, uses a refined grammar to generate an n-best list of parse candidates. Charniak and Johnson (2005) extended this pipeline with a discriminative maximum entropy model to rerank the n-best parse candidates, deriving a significant benefit from the richer model employed by the reranker.

For our experiments, we modified the parser[1] to allow us to optionally provide base-phrase trees to constrain the first stage of parsing.

[1] ftp://ftp.cs.brown.edu/pub/nlparser/

Parser		Base Phrases	Shallow Phrases
Charniak	parser-best	91.9	94.4
	reranker-best	92.8	94.8
Finite-state shallow parser		91.7	94.3

Table 1: F-scores on WSJ section 24 of output from two parsers on the similar tasks of base-phrase parsing and shallow-phrase parsing. For evaluation, base and shallow phrases are extracted from the Charniak/Johnson full-parse output.

2.1 Base Phrases

Following Ratnaparkhi (1999), we define a *base phrase* as any parse node with only preterminal children. Unlike the shallow phrases defined for the CoNLL-2000 Shared Task (Tjong Kim Sang and Buchholz, 2000), base phrases correspond directly to constituents that appear in full parses, and hence can provide a straightforward constraint on edges within a chart parser. In contrast, shallow phrases collapse certain non-constituents—such as auxiliary chains—into a single phrase, and hence are not directly applicable as constraints on a chart parser.

We have two methods for deriving base-phrase annotations for a string. First, we trained a finite-state shallow parser on base phrases extracted from the Penn Wall St. Journal (WSJ) Treebank (Marcus et al., 1993). The treebank trees are pre-processed identically to the procedure for training the Charniak parser, e.g., empty nodes and function tags are removed. The shallow parser is trained using the perceptron algorithm, with a feature set nearly identical to that from Sha and Pereira (2003), and achieves comparable performance to that paper. See Hollingshead et al. (2005) for more details. Second, base phrases can be extracted from the full-parse output of the Charniak and Johnson (2005) reranker, via a simple script to extract nodes with only preterminal children.

Table 1 shows these systems' bracketing accuracy on both the base-phrase and shallow parsing tasks for WSJ section 24; each system was trained on WSJ sections 02-21. From this table we can see that base phrases are substantially more difficult than shallow phrases to annotate. Output from the finite-state shallow parser is roughly as accurate as output extracted from the Charniak parser-best trees, though a fair amount below output extracted from the reranker-best trees.

In addition to using base phrase constraints from these two sources independently, we also looked at

combining the predictions of both to obtain more reliable constraints. We next present a method of combining output from multiple parsers based on combined precision and recall optimization.

2.2 Combining Parser n-best Lists

In order to select high-likelihood constraints for the pipeline, we may want to extract annotations with high levels of agreement ("consensus hypotheses") between candidates. In addition, we may want to favor precision over recall to avoid erroneous constraints within the pipeline as much as possible. Here we discuss how a technique presented in Goodman's thesis (1998) can be applied to do this.

We will first present this within a general chart parsing approach, then move to how we use it for n-best lists. Let $\mathcal{T}$ be the set of trees for a particular input, and let a parse $T \in \mathcal{T}$ be considered as a set of labeled spans. Then, for all labeled spans $X \in T$, we can calculate the posterior probability $\gamma(X)$ as follows:

$$\gamma(X) \quad = \quad \sum_{T \in \mathcal{T}} \frac{\mathrm{P}(T)[\![X \in T]\!]}{\sum_{T' \in \mathcal{T}} \mathrm{P}(T')} \qquad (1)$$

where $[\![X \in T]\!] = \begin{cases} 1 & \text{if } X \in T \\ 0 & \text{otherwise.} \end{cases}$

Goodman (1996; 1998) presents a method for using the posterior probability of constituents to maximize the expected labeled recall of binary branching trees, as follows:

$$\widehat{T} \quad = \quad \operatorname*{argmax}_{T \in \mathcal{T}} \sum_{X \in T} \gamma(X) \qquad (2)$$

Essentially, find the tree with the maximum sum of the posterior probabilities of its constituents. This is done by computing the posterior probabilities of constituents in a chart, typically via the Inside-Outside algorithm (Baker, 1979; Lari and Young, 1990), followed by a final CYK-like pass to find the tree maximizing the sum.

For non-binary branching trees, where precision and recall may differ, Goodman (1998, Ch.3) proposes the following combined metric for balancing precision and recall:

$$\widehat{T} \quad = \quad \operatorname*{argmax}_{T \in \mathcal{T}} \sum_{X \in T} (\gamma(X) - \lambda) \qquad (3)$$

where λ ranges from 0 to 1. Setting λ=0 is equivalent to Eq. 2 and thus optimizes recall, and setting λ=1 optimizes precision; Appendix 5 at the end of this paper presents brief derivations of these metrics.[2] Thus, λ functions as a mixing factor to balance recall and precision.

This approach also gives us a straightforward way to combine n-best outputs of multiple systems. To do this, we construct a chart of the constituents in the trees from the n-best lists, and allow any combination of constituents that results in a tree – even one with no internal structure. In such a way, we can produce trees that only include a small number of high-certainty constituents, and leave the remainder of the string unconstrained, even if such trees were not candidates in the original n-best lists.

For simplicity, we will here discuss the combination of two n-best lists, though it generalizes in the obvious way to an arbitrary number of lists. Let $\mathcal{T}$ be the union of the two n-best lists. For all trees $T \in \mathcal{T}$, let $\mathrm{P}_1(T)$ be the probability of T in the first n-best list, and $\mathrm{P}_2(T)$ the probability of T in the second n-best list. Then, we define $\mathrm{P}(T)$ as follows:

$$\mathrm{P}(T) \quad = \quad \alpha \frac{\mathrm{P}_1(T)}{\sum_{T' \in \mathcal{T}} \mathrm{P}_1(T')} + \frac{\mathrm{P}_2(T)}{\sum_{T' \in \mathcal{T}} \mathrm{P}_2(T')} \qquad (4)$$

where the parameter α dictates the relative weight of P_1 versus P_2 in the combination.[3]

For this paper, we combined two n-best lists of base-phrase trees. Although there is no hierarchical structure in base-phrase annotations, they can be represented as flat trees, as shown in Fig. 2(a). We constructed a chart from the two lists being combined, using Eq. 4 to define $\mathrm{P}(T)$ in Eq. 1. We wish to consider every possible combination of the base phrases, so for the final CYK-like pass to find the argmax tree, we included rules for attaching each preterminal directly to the root of the tree, in addition to rules permitting any combination of hypothesized base phrases.

Consider the trees in Fig. 2. Figure 2(a) is a shallow parse with three NP base phrases; Figure 2(b) is the same parse where the ROOT production has been binarized for the final CYK-like pass, which requires binary productions. If we include productions of the form 'ROOT → X ROOT' and 'ROOT → X Y' for all non-terminals X and Y (including POS tags), then any tree-structured combination of base phrases hypothesized in either n-

[2] Our notation differs slightly from that in Goodman (1998), though the approaches are formally equivalent.

[3] Note that P_1 and P_2 are normalized in eq. 4, and thus are not required to be true probabilities. In turn, P is normalized when used in eq. 1, such that the posterior probability γ is a true probability. Hence P need not be normalized in eq. 4.

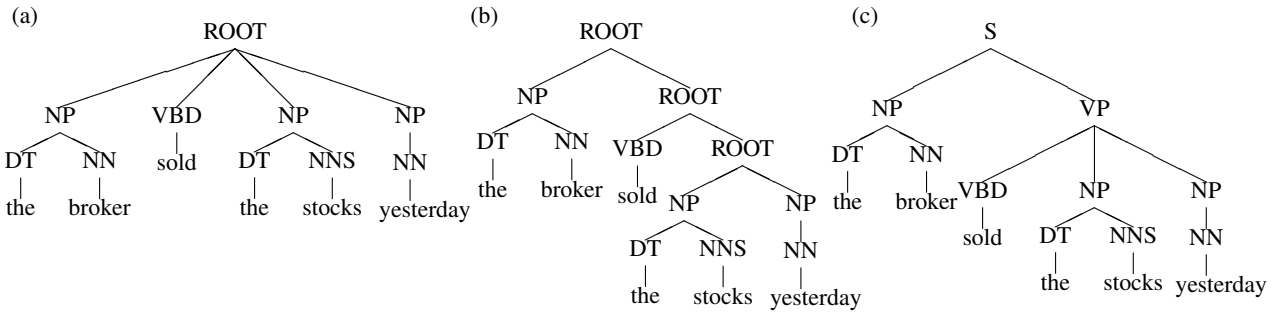


Figure 2: Base-phrase trees (a) as produced for an n-best list and (b) after root-binarization for n-best list combination. Full-parse tree (c) consistent with constraining base-phrase tree (a).

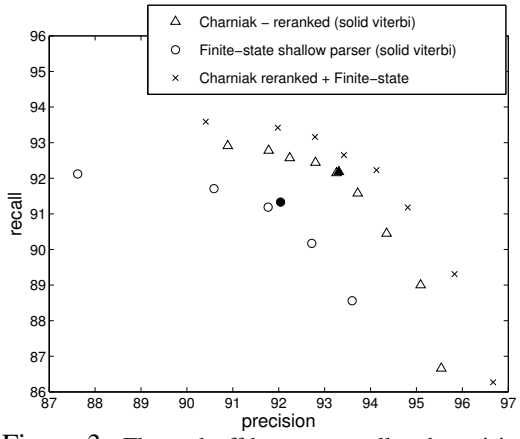


Figure 3: The tradeoff between recall and precision using a range of λ values (Eq. 3) to select high-probability annotations from an n-best list. Results are shown on 50-best lists of base-phrase parses from two parsers, and on the combination of the two lists.

best list is allowed, including the one with no base phrases at all. Note that, for the purpose of finding the argmax tree in Eq. 3, we only sum the posterior probabilities of base-phrase constituents, and not the ROOT symbol or POS tags.

Figure 3 shows the results of performing this combined precision/recall optimization on three separate n-best lists: the 50-best list of base-phrase trees extracted from the full-parse output of the Charniak and Johnson (2005) reranker; the 50-best list output by the Hollingshead et al. (2005) finite-state shallow parser; and the weighted combination of the two lists at various values of λ in Eq. 3. For the combination, we set α=2 in Eq. 4, with the Charniak and Johnson (2005) reranker providing P_1, effectively giving the reranker twice the weight of the shallow parser in determining the posteriors. The shallow parser has perceptron scores as weights, and the distribution of these scores after a softmax normalization was too peaked to be of utility, so we used the normalized reciprocal rank of each candidate as P_2 in Eq. 4.

We point out several details in these results. First, using this method does not result in an F-measure improvement over the Viterbi-best base-phrase parses (shown as solid symbols in the graph) for either the reranker or shallow parser. Also, using this model effects a greater improvement in precision than in recall, which is unsurprising with these non-hierarchical annotations; unlike full parsing (where long sequences of unary productions can improve recall arbitrarily), in base-phrase parsing, any given span can have only one non-terminal. Finally, we see that the combination of the two n-best lists improves over either list in isolation.

3 Experimental Setup

For our experiments we constructed a simple parsing pipeline, shown in Fig. 4. At the core of the pipeline is the Charniak and Johnson (2005) coarse-to-fine parser and MaxEnt reranker, described in Sec. 2. The parser constitutes the first and second stages of our pipeline, and the reranker the final stage. Following Charniak and Johnson (2005), we set the parser to output 50-best parses for all experiments described here. We constrain only the first stage of the parser: during chart construction, we disallow any constituents that conflict with the constraints, as described in detail in the next section.

3.1 Parser Constraints

We use base phrases, as defined in Sec. 2.1, to constrain the first stage of our parsing pipeline. Under these constraints, full parses must be consistent with the base-phrase tree provided as input to the parser, i.e., any valid parse must contain all of the base-phrase constituents in the constraining tree. The full-parse tree in Fig. 2(c), for example, is consistent with the base-phrase tree in Fig. 2(a).

Implementing these constraints in a parser is straightforward, one of the advantages of using base phrases as constraints. Since the internal structure of base phrases is, by definition, limited to preterminal children, we can constrain the entire parse by constraining the parents of the appropriate preterminal nodes. For any preterminal that occurs within the span of a constraining base phrase, the only valid parent is a node matching both the *span* (start and end points) and the *label* of the provided base

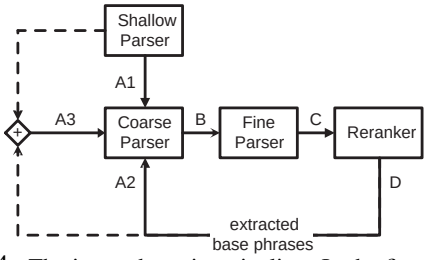


Figure 4: The iterated parsing pipeline. In the first iteration, the coarse parser may be either unconstrained, or constrained by base phrases from the shallow parser (A1). In the second iteration, base phrase constraints may be extracted either from reranker output (A2) or from a weighted combination of shallow parser output and reranker output (A3). Multiple sets of n-best parses, as output by the coarse-to-fine parser under different constraint conditions, may be joined in a set union (C).

phrase. All other proposed parent-nodes are rejected. In such a way, for any parse to cover the entire string, it would have to be consistent with the constraining base-phrase tree.

Words that fall outside of any base-phrase constraint are unconstrained in how they attach within the parse; hence, a base-phrase tree with few words covered by base-phrase constraints will result in a larger search space than one with many words covered by base phrases. We also put no restrictions on the preterminal labels, even within the base phrases. We normalized for punctuation. If the parser fails to find a valid parse with the constraints, then we lift the constraints and allow any parse constituent originally proposed by the first-stage parser.

3.2 Experimental Conditions

Our experiments will demonstrate the effects of constraining the Charniak parser under several different conditions. The baseline system places no constraints on the parser. The remaining experimental conditions each consider one of three possible sources of the base phrase constraints: (1) the base phrases output by the finite-state shallow parser; (2) the base phrases extracted from output of the reranker; and (3) a combination of the output from the shallow parser and the reranker, which is produced using the techniques outlined in Sec. 2.2. Constraints are enforced as described in Sec. 3.1.

Unconstrained For our baseline system, we run the Charniak and Johnson (2005) parser and reranker with default parameters. The parser is provided with treebank-tokenized text and, as mentioned previously, outputs 50-best parse candidates to the reranker.

FS-constrained The FS-constrained condition provides a comparison point of non-iterated constraints. Under this condition, the one-best base-phrase tree output by the finite-state shallow parser is input as a constraint to the Charniak parser. We run the parser and reranker as before, under constraints from the shallow parser. The accuracy of the constraints used under this condition is shown in the first row of Table 2. Note that this condition is *not* an instance of pipeline iteration, but is included to show the performance levels that can be achieved without iteration.

System	LR	LP	F
Finite-state shallow parser	91.3	92.0	91.7
Charniak reranker-best	92.2	93.3	92.8
Combination (λ=0.5)	92.2	94.1	93.2
Combination (λ=0.9)	81.0	97.4	88.4

Table 2: Labeled recall (LR), precision (LP), and F-scores on WSJ section 24 of base-phrase trees produced by the three possible sources of constraints.

Reranker-constrained We will use the reranker-constrained condition to examine the effects of pipeline iteration, with no input from other models outside the pipeline. We take the reranker-best full-parse output under the condition of unconstrained search, and extract the corresponding base-phrase tree. We run the parser and reranker as before, now with constraints from the reranker. The accuracy of the constraints used under this condition is shown in the second row of Table 2.

Combo-constrained The combo-constrained conditions are designed to compare the effects of generating constraints with different combination parameterizations, i.e., different λ parameters in Eq. 3. For this experimental condition, we extract base-phrase trees from the n-best full-parse trees output by the reranker. We combine this list with the n-best list output by the finite-state shallow parser, exactly as described in Sec. 2.2, again with the reranker providing P_1 and α=2 in Eq. 4. We examined a range of operating points from λ=0.4 to λ=0.9, and report two points here (λ=0.5 and λ=0.9), which represent the highest overall accuracy and the highest precision, respectively, as shown in Table 2.

Constrained and Unconstrained Union When iterating this pipeline, the original n-best list of full parses output from the unconstrained parser is available at no additional cost, and our final set of experimental conditions investigate taking the union of constrained and unconstrained n-best lists. The imposed constraints can result in candidate sets that are largely (or completely) disjoint from the unconstrained sets, and it may be that the unconstrained set is in many cases superior to the constrained set.

Constraints	Parser-best	Reranker-best	Oracle-best	# Candidates
Baseline (Unconstrained, 50-best)	88.92	90.24	95.95	47.9
FS-constrained	88.44	89.50	94.10	46.2
Reranker-constrained	89.60	90.46	95.07	46.9
Combo-constrained (λ=0.5)	89.81	90.74	95.41	46.3
Combo-constrained (λ=0.9)	89.34	90.43	95.91	47.5

Table 3: Full-parse F-scores on WSJ section 24. The unconstrained search (first row) provides a baseline comparison for the effects of constraining the search space. The last four rows demonstrate the effect of various constraint conditions.

Even our high-precision constraints did not reach 100% precision, attesting to the fact that there was some error in all constrained conditions. By constructing the union of the two n-best lists, we can take advantage of the new constrained candidate set without running the risk that the constraints have resulted in a worse n-best list. Note that the parser probabilities are produced from the same model in both passes, and are hence directly comparable.

The output of the second pass of the pipeline could be used to constrain a third pass, for multiple pipeline iterations. However, we found that further iterations provided no additional improvements.

3.3 Data

Unless stated otherwise, all reported results will be F-scores on WSJ section 24 of the Penn WSJ Treebank, which was our development set. Training data was WSJ sections 02-21, with section 00 as held-out data. Crossfold validation (20-fold with 2,000 sentences per fold) was used to train the reranker for every condition. Evaluation was performed using `evalb` under standard parameterizations. WSJ section 23 was used only for final testing.

4 Results & Discussion

We evaluate the one-best parse candidates before and after reranking (*parser-best* and *reranker-best*, respectively). We additionally provide the best-possible F-score in the n-best list (*oracle-best*) and the number of unique candidates in the list.

Table 3 presents trials showing the effect of constraining the parser under various conditions. Constraining the parser to the base phrases produced by the finite-state shallow parser (FS-constrained) hurts performance by half a point. Constraining the parser to the base phrases produced by the reranker, however, provides a 0.7 percent improvement in the parser-best accuracy, and a 0.2 percent improvement after reranking. Combining the two base-phrase n-best lists to derive the constraints provides further improvements when λ=0.5, to a total improvement of 0.9 and 0.5 percent over parser-best and reranker-best accuracy, respectively. Performance degrades at λ=0.9 relative to λ=0.5, indicating that, even at a lower precision, more constraints are beneficial.

The oracle rate decreases under all of the constrained conditions as compared to the baseline, demonstrating that the parser was prevented from finding some of the best solutions that were originally found. However, the improvement in F-score shows that the constraints assisted the parser in achieving high-quality solutions despite this degraded oracle accuracy of the lists.

Table 4 shows the results when taking the union of the constrained and unconstrained lists prior to reranking. Several interesting points can be noted in this table. First, despite the fact that the FS-constrained condition hurts performance in Table 3, the union provides a 0.5 percent improvement over the baseline in the parser-best performance. This indicates that, in some cases, the Charniak parser is scoring parses in the constrained set higher than in the unconstrained set, which is evidence of search errors in the unconstrained condition. One can see from the number of candidates that the FS-constrained condition provides the set of candidates most disjoint from the original unconstrained parser, leading to the largest number of candidates in the union. Surprisingly, even though this set provided the highest parser-best F-score of all of the union sets, it did not lead to significant overall improvements after reranking.

In all other conditions, taking the union decreases the parser-best accuracy when compared to the corresponding constrained output, but improves the reranker-best accuracy in all but the combo-constrained λ=0.9 condition. One explanation for the lower performance at λ=0.9 versus λ=0.5 is seen in the number of candidates, about 7.5 fewer than in the λ=0.5 condition. There are fewer constraints in the high-precision condition, so the resulting n-best lists do not diverge as much from the original lists, leading to less diversity in their union.

The gains in performance should not be attributed to increasing the number of candidates nor to allow-

Constraints	Parser-best	Reranker-best	Oracle-best	# Candidates
Baseline (Unconstrained, 50-best)	88.92	90.24	95.95	47.9
Unconstrained ∪ FS-constrained	89.39	90.27	96.61	74.9
Unconstrained ∪ Reranker-constrained	89.23	90.59	96.48	70.3
Unconstrained ∪ Combo (λ=0.5)	89.28	90.78	96.53	69.7
Unconstrained ∪ Combo (λ=0.9)	89.03	90.44	96.40	62.1
Unconstrained (100-best)	88.82	90.13	96.38	95.2
Unconstrained (50-best, beam×2)	89.01	90.45	96.13	48.1

Table 4: Full-parse F-scores on WSJ section 24 after taking the set union of unconstrained and constrained parser output under the 4 different constraint conditions. Also, F-score for 100-best parses, and 50-best parses with an increased beam threshold, output by the Charniak parser under the unconstrained condition.

Constraints	F-score
Baseline (Unconstrained, 50-best)	91.06
Unconstrained ∪ Combo (λ=0.5)	91.48

Table 5: Full-parse F-scores on WSJ section 23 for our best-performing system on WSJ section 24. The 0.4 percent F-score improvement is significant at $p < 0.001$.

ing the parser more time to generate the parses. The penultimate row in Table 4 shows the results with 100-best lists output in the unconstrained condition, which does not improve upon the 50-best performance, despite an improved oracle F-score. Since the second iteration through the parsing pipeline clearly increases the overall processing time by a factor of two, we also compare against output obtained by doubling the coarse-parser's beam threshold. The last row in Table 4 shows that the increased threshold yields an insignificant improvement over the baseline, despite a very large processing burden.

We applied our best-performing model (Unconstrained ∪ Combo, λ=0.5) to the test set, WSJ section 23, for comparison against the baseline system. Table 5 shows a 0.4 percent F-score improvement over the baseline for that section, which is statistically significant at $p < 0.001$, using the stratified shuffling test (Yeh, 2000).

5 Conclusion & Future Work

In summary, we have demonstrated that pipeline iteration can be useful in improving system performance, by constraining early stages of the pipeline with output derived from later stages. While the current work made use of a particular kind of constraint—base phrases—many others could be extracted as well. Preliminary results extending the work presented in this paper show parser accuracy improvements from pipeline iteration when using constraints based on an unlabeled partial bracketing of the string. Higher-level phrase segmentations or fully specified trees over portions of the string might also prove to be effective constraints. The techniques shown here are by no means limited to parsing pipelines, and could easily be applied to other tasks making use of pipeline architectures.

Acknowledgments

Thanks to Martin Jansche for useful discussions on topics related to this paper. The first author of this paper was supported under an NSF Graduate Research Fellowship. In addition, this research was supported in part by NSF Grant #IIS-0447214. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF.

References

J.K. Baker. 1979. Trainable grammars for speech recognition. In *Speech Communication papers for the 97th Meeting of the Acoustical Society of America*.

D. Blaheta and E. Charniak. 1999. Automatic compensation for parser figure-of-merit flaws. In *Proceedings of the 37th Annual Meeting of ACL*, pages 513–518.

S. Caraballo and E. Charniak. 1998. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24(2):275–298.

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

E. Charniak, S. Goldwater, and M. Johnson. 1998. Edge-based best-first chart parsing. In *Proceedings of the 6th Workshop for Very Large Corpora*, pages 127–133.

E. Charniak, M. Johnson, M. Elsner, J.L. Austerweil, D. Ellis, S.R. Iyangar, J. Moore, M.T. Pozar, C. Hill, T.Q. Vu, and I. Haxton. 2006. Multi-level course-to-fine PCFG parsing. In *Proceedings of the HLT-NAACL Annual Meeting*, pages 168–175.

E. Charniak. 2000. A Maximum-Entropy-inspired parser. In *Proceedings of the 1st Annual Meeting of NAACL and 6th Conference on ANLP*, pages 132–139.

J.R. Finkel, C.D. Manning, and A.Y. Ng. 2006. Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines. In *Proceedings of EMNLP*, pages 618–626.

J. Fiscus. 1997. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER). In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*.

V. Goel, S. Kumar, and W. Byrne. 2000. Segmental minimum Bayes-risk ASR voting strategies. In *Proceedings of ICSLP*, pages 139–142.

J. Goodman. 1996. Parsing algorithms and metrics. In *Proceedings of the 34th Annual Meeting of ACL*, pages 177–183.

J. Goodman. 1998. *Parsing inside-out*. Ph.D. thesis, Harvard University.

K. Hall and M. Johnson. 2004. Attention shifting for parsing speech. In *Proceedings of the 42nd Annual Meeting of ACL*, pages 40–46.

K. Hollingshead, S. Fisher, and B. Roark. 2005. Comparing and combining finite-state and context-free parsers. In *Proceedings of HLT-EMNLP*, pages 787–794.

K. Lari and S.J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4(1):35–56.

M.P. Marcus, M.A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19:314–330.

D. McClosky, E. Charniak, and M. Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of COLING-ACL*, pages 337–344.

F.J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29.

A. Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3):151–175.

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the HLT-NAACL Annual Meeting*, pages 134–141.

E.F. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL*, pages 127–132.

A. Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International COLING*, pages 947–953.

Appendix A Combined Precision/Recall Decoding

Recall that $\mathcal{T}$ is the set of trees for a particular input, and each $T \in \mathcal{T}$ is considered as a set of labeled spans. For all labeled spans $X \in T$, we can calculate the posterior probability $\gamma(X)$ as follows:

$$\gamma(X) \;=\; \sum_{T \in \mathcal{T}} \frac{\mathrm{P}(T)[\![X \in T]\!]}{\sum_{T' \in \mathcal{T}} \mathrm{P}(T')}$$

where $[\![X \in T]\!] = \begin{cases} 1 & \text{if } X \in T \\ 0 & \text{otherwise.} \end{cases}$

If τ is the reference tree, the labeled precision (LP) and labeled recall (LR) of a T relative to τ are defined as

$$\mathrm{LP} \;=\; \frac{|T \cap \tau|}{|T|} \qquad \mathrm{LR} \;=\; \frac{|T \cap \tau|}{|\tau|}$$

where $|T|$ denotes the size of the set T.

A metric very close to LR is $|T \cap \tau|$, the number of nodes in common between the tree and the reference tree. To maximize the expected value ($\mathcal{E}$) of this metric, we want to find the tree $\widehat{T}$ as follows:

$$\begin{aligned}
\widehat{T} &= \operatorname*{argmax}_{T \in \mathcal{T}} \mathcal{E}\left[|T \bigcap \tau|\right] \\
&= \operatorname*{argmax}_{T \in \mathcal{T}} \sum_{T' \in \mathcal{T}} \frac{\mathrm{P}(T')\left[|T \bigcap T'|\right]}{\sum_{T'' \in \mathcal{T}} \mathrm{P}(T'')} \\
&= \operatorname*{argmax}_{T \in \mathcal{T}} \sum_{T' \in \mathcal{T}} \frac{\mathrm{P}(T') \sum_{X \in T} [\![X \in T']\!]}{\sum_{T'' \in \mathcal{T}} \mathrm{P}(T'')} \\
&= \operatorname*{argmax}_{T \in \mathcal{T}} \sum_{X \in T} \sum_{T' \in \mathcal{T}} \frac{\mathrm{P}(T')[\![X \in T']\!]}{\sum_{T'' \in \mathcal{T}} \mathrm{P}(T'')} \\
&= \operatorname*{argmax}_{T \in \mathcal{T}} \sum_{X \in T} \gamma(X) \qquad (5)
\end{aligned}$$

This exactly maximizes the expected LR in the case of binary branching trees, and is closely related to LR for non-binary branching trees. Similar to maximizing the expected number of matching nodes, we can minimize the expected number of non-matching nodes, for a metric related to LP:

$$\begin{aligned}
\widehat{T} &= \operatorname*{argmin}_{T \in \mathcal{T}} \mathcal{E}\left[|T| - |T \bigcap \tau|\right] \\
&= \operatorname*{argmax}_{T \in \mathcal{T}} \mathcal{E}\left[|T \bigcap \tau| - |T|\right] \\
&= \operatorname*{argmax}_{T \in \mathcal{T}} \sum_{T' \in \mathcal{T}} \frac{\mathrm{P}(T')\left[|T \bigcap T'| - |T|\right]}{\sum_{T'' \in \mathcal{T}} \mathrm{P}(T'')} \\
&= \operatorname*{argmax}_{T \in \mathcal{T}} \sum_{T' \in \mathcal{T}} \frac{\mathrm{P}(T') \sum_{X \in T} ([\![X \in T']\!] - 1)}{\sum_{T'' \in \mathcal{T}} \mathrm{P}(T'')} \\
&= \operatorname*{argmax}_{T \in \mathcal{T}} \sum_{X \in T} \sum_{T' \in \mathcal{T}} \frac{\mathrm{P}(T')([\![X \in T']\!] - 1)}{\sum_{T'' \in \mathcal{T}} \mathrm{P}(T'')} \\
&= \operatorname*{argmax}_{T \in \mathcal{T}} \sum_{X \in T} (\gamma(X) - 1) \qquad (6)
\end{aligned}$$

Finally, we can combine these two metrics in a linear combination. Let λ be a mixing factor from 0 to 1. Then we can optimize the weighted sum:

$$\begin{aligned}
\widehat{T} &= \operatorname*{argmax}_{T \in \mathcal{T}} \mathcal{E}\left[(1-\lambda)|T \bigcap \tau| + \lambda(|T \bigcap \tau| - |T|)\right] \\
&= \operatorname*{argmax}_{T \in \mathcal{T}} (1-\lambda)\mathcal{E}\left[|T \bigcap \tau|\right] + \lambda\mathcal{E}\left[|T \bigcap \tau| - |T|\right] \\
&= \operatorname*{argmax}_{T \in \mathcal{T}} \left[(1-\lambda) \sum_{X \in T} \gamma(X)\right] + \left[\lambda \sum_{X \in T} (\gamma(X) - 1)\right] \\
&= \operatorname*{argmax}_{T \in \mathcal{T}} \sum_{X \in T} (\gamma(X) - \lambda) \qquad (7)
\end{aligned}$$

The result is a combined metric for balancing precision and recall. Note that, if λ=0, Eq. 7 is the same as Eq. 5 and thus maximizes the LR metric; and if λ=1, Eq. 7 is the same as Eq. 6 and thus maximizes the LP metric.

Learning Synchronous Grammars for Semantic Parsing with Lambda Calculus

Yuk Wah Wong and Raymond J. Mooney
Department of Computer Sciences
The University of Texas at Austin
{ywwong,mooney}@cs.utexas.edu

Abstract

This paper presents the first empirical results to our knowledge on learning synchronous grammars that generate logical forms. Using statistical machine translation techniques, a semantic parser based on a synchronous context-free grammar augmented with λ-operators is learned given a set of training sentences and their correct logical forms. The resulting parser is shown to be the best-performing system so far in a database query domain.

1 Introduction

Originally developed as a theory of compiling programming languages (Aho and Ullman, 1972), synchronous grammars have seen a surge of interest recently in the *statistical machine translation* (SMT) community as a way of formalizing syntax-based translation models between natural languages (NL). In generating multiple parse trees in a single derivation, synchronous grammars are ideal for modeling syntax-based translation because they describe not only the hierarchical structures of a sentence and its translation, but also the exact correspondence between their sub-parts. Among the grammar formalisms successfully put into use in syntax-based SMT are synchronous context-free grammars (SCFG) (Wu, 1997) and synchronous tree-substitution grammars (STSG) (Yamada and Knight, 2001). Both formalisms have led to SMT systems whose performance is state-of-the-art (Chiang, 2005; Galley et al., 2006).

Synchronous grammars have also been used in other NLP tasks, most notably *semantic parsing*, which is the construction of a complete, formal *meaning representation* (MR) of an NL sentence. In our previous work (Wong and Mooney, 2006), semantic parsing is cast as a machine translation task, where an SCFG is used to model the translation of an NL into a formal meaning-representation language (MRL). Our algorithm, WASP, uses statistical models developed for syntax-based SMT for lexical learning and parse disambiguation. The result is a robust semantic parser that gives good performance in various domains. More recently, we show that our SCFG-based parser can be inverted to produce a state-of-the-art NL generator, where a formal MRL is translated into an NL (Wong and Mooney, 2007).

Currently, the use of learned synchronous grammars in semantic parsing and NL generation is limited to simple MRLs that are free of logical variables. This is because grammar formalisms such as SCFG do not have a principled mechanism for handling logical variables. This is unfortunate because most existing work on computational semantics is based on predicate logic, where logical variables play an important role (Blackburn and Bos, 2005). For some domains, this problem can be avoided by transforming a logical language into a variable-free, functional language (e.g. the GEOQUERY functional query language in Wong and Mooney (2006)). However, development of such a functional language is non-trivial, and as we will see, logical languages can be more appropriate for certain domains.

On the other hand, most existing methods for mapping NL sentences to logical forms involve substantial hand-written components that are difficult to maintain (Joshi and Vijay-Shanker, 2001; Bayer et al., 2004; Bos, 2005). Zettlemoyer and Collins (2005) present a statistical method that is consider-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 960–967,
Prague, Czech Republic, June 2007.

ably more robust, but it still relies on hand-written rules for lexical acquisition, which can create a performance bottleneck.

In this work, we show that methods developed for SMT can be brought to bear on tasks where logical forms are involved, such as semantic parsing. In particular, we extend the WASP semantic parsing algorithm by adding variable-binding λ-operators to the underlying SCFG. The resulting synchronous grammar generates logical forms using λ-calculus (Montague, 1970). A semantic parser is learned given a set of sentences and their correct logical forms using SMT methods. The new algorithm is called λ-WASP, and is shown to be the best-performing system so far in the GEOQUERY domain.

2 Test Domain

In this work, we mainly consider the GEOQUERY domain, where a query language based on Prolog is used to query a database on U.S. geography (Zelle and Mooney, 1996). The query language consists of logical forms augmented with meta-predicates for concepts such as *smallest* and *count*. Figure 1 shows two sample logical forms and their English glosses. Throughout this paper, we use the notation $x_1, x_2, \ldots$ for logical variables.

Although Prolog logical forms are the main focus of this paper, our algorithm makes minimal assumptions about the target MRL. The only restriction on the MRL is that it be defined by an unambiguous context-free grammar (CFG) that divides a logical form into subformulas (and terms into subterms). Figure 2(a) shows a sample parse tree of a logical form, where each CFG production corresponds to a subformula.

3 The Semantic Parsing Algorithm

Our work is based on the WASP semantic parsing algorithm (Wong and Mooney, 2006), which translates NL sentences into MRs using an SCFG. In WASP, each SCFG production has the following form:

$$A \rightarrow \langle \alpha, \beta \rangle \tag{1}$$

where α is an NL phrase and β is the MR translation of α. Both α and β are strings of terminal and non-terminal symbols. Each non-terminal in α appears in β exactly once. We use indices to show the correspondence between non-terminals in α and β. All derivations start with a pair of co-indexed start symbols, $\langle S_{\boxed{1}}, S_{\boxed{1}} \rangle$. Each step of a derivation involves the rewriting of a pair of co-indexed non-terminals by the same SCFG production. The yield of a derivation is a pair of terminal strings, $\langle \mathbf{e}, \mathbf{f} \rangle$, where $\mathbf{e}$ is an NL sentence and $\mathbf{f}$ is the MR translation of $\mathbf{e}$. For convenience, we call an SCFG production a *rule* throughout this paper.

While WASP works well for target MRLs that are free of logical variables such as CLANG (Wong and Mooney, 2006), it cannot easily handle various kinds of logical forms used in computational semantics, such as predicate logic. The problem is that WASP lacks a principled mechanism for handling logical variables. In this work, we extend the WASP algorithm by adding a variable-binding mechanism based on λ-calculus, which allows for compositional semantics for logical forms.

This work is based on an extended version of SCFG, which we call λ-SCFG, where each rule has the following form:

$$A \rightarrow \langle \alpha, \lambda x_1 \ldots \lambda x_k.\beta \rangle \tag{2}$$

where α is an NL phrase and β is the MR translation of α. Unlike (1), β is a string of terminals, non-terminals, and *logical variables*. The variable-binding operator λ binds occurrences of the logical variables $x_1, \ldots, x_k$ in β, which makes $\lambda x_1 \ldots \lambda x_k.\beta$ a λ-*function* of arity k. When applied to a list of arguments, $(x_{i_1}, \ldots, x_{i_k})$, the λ-function gives $\beta\sigma$, where σ is a substitution operator, $\{x_1/x_{i_1}, \ldots, x_k/x_{i_k}\}$, that replaces all bound occurrences of x_j in β with x_{i_j}. If any of the arguments x_{i_j} appear in β as a free variable (i.e. not bound by any λ), then those free variables in β must be renamed before function application takes place.

Each non-terminal A_j in β is followed by a list of arguments, $\mathbf{x}_j = (x_{j_1}, \ldots, x_{j_{k_j}})$. During parsing, A_j must be rewritten by a λ-function f_j of arity k_j. Like SCFG, a derivation starts with a pair of co-indexed start symbols and ends when all non-terminals have been rewritten. To compute the yield of a derivation, each f_j is applied to its corresponding arguments $\mathbf{x}_j$ to obtain an MR string free of λ-operators with logical variables properly named.

(a) `answer(`x_1`,smallest(`x_2`,(state(`x_1`),area(`x_1`,`x_2`))))`
What is the smallest state by area?

(b) `answer(`x_1`,count(`x_2`,(city(`x_2`),major(`x_2`),loc(`x_2`,`x_3`),next_to(`x_3`,`x_4`),state(`x_3`),`
`equal(`x_4`,stateid(texas)))))`
How many major cities are in states bordering Texas?

Figure 1: Sample logical forms in the GEOQUERY domain and their English glosses.

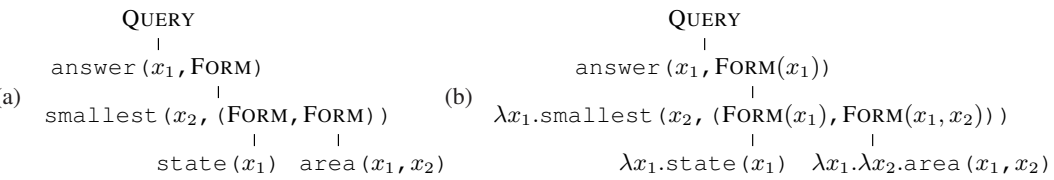


Figure 2: Parse trees of the logical form in Figure 1(a).

As a concrete example, Figure 2(b) shows an MR parse tree that corresponds to the English parse, [*What is the* [*smallest* [*state*] [*by area*]]], based on the λ-SCFG rules in Figure 3. To compute the yield of this MR parse tree, we start from the leaf nodes: apply $\lambda x_1.\texttt{state}(x_1)$ to the argument (x_1), and $\lambda x_1.\lambda x_2.\texttt{area}(x_1,x_2)$ to the arguments (x_1, x_2). This results in two MR strings: $\texttt{state}(x_1)$ and $\texttt{area}(x_1,x_2)$. Substituting these MR strings for the FORM non-terminals in the parent node gives the λ-function $\lambda x_1.\texttt{smallest}(x_2,\texttt{(state}(x_1)\texttt{,area}(x_1,x_2)\texttt{))}$. Applying this λ-function to (x_1) gives the MR string $\texttt{smallest}(x_2,\texttt{(state}(x_1)\texttt{,area}(x_1,x_2)\texttt{))}$. Substituting this MR string for the FORM non-terminal in the grandparent node in turn gives the logical form in Figure 1(a). This is the yield of the MR parse tree, since the root node of the parse tree is reached.

3.1 Lexical Acquisition

Given a set of training sentences paired with their correct logical forms, $\{\langle \mathbf{e}_i, \mathbf{f}_i \rangle\}$, the main learning task is to find a λ-SCFG, G, that covers the training data. Like most existing work on syntax-based SMT (Chiang, 2005; Galley et al., 2006), we construct G using rules extracted from word alignments. We use the $K = 5$ most probable word alignments for the training set given by GIZA++ (Och and Ney, 2003), with variable names ignored to reduce sparsity. Rules are then extracted from each word alignment as follows.

To ground our discussion, we use the word alignment in Figure 4 as an example. To represent the logical form in Figure 4, we use its linearized parse—a list of MRL productions that generate the logical form, in top-down, left-most order (cf. Figure 2(a)). Since the MRL grammar is unambiguous, every logical form has a unique linearized parse. We assume the alignment to be n-to-1, where each word is linked to at most one MRL production.

Rules are extracted in a bottom-up manner, starting with MRL productions at the leaves of the MR parse tree, e.g. FORM $\rightarrow \texttt{state}(x_1)$ in Figure 2(a). Given an MRL production, $A \rightarrow \beta$, a rule $A \rightarrow \langle \alpha, \lambda x_{i_1} \ldots \lambda x_{i_k}.\beta \rangle$ is extracted such that: (1) α is the NL phrase linked to the MRL production; (2) $x_{i_1}, \ldots, x_{i_k}$ are the logical variables that appear in β and outside the current leaf node in the MR parse tree. If $x_{i_1}, \ldots, x_{i_k}$ were not bound by λ, they would become free variables in β, subject to renaming during function application (and therefore, invisible to the rest of the logical form). For example, since x_1 is an argument of the `state` predicate as well as `answer` and `area`, x_1 must be bound (cf. the corresponding tree node in Figure 2(b)). The rule extracted for the `state` predicate is shown in Figure 3.

The case for the internal nodes of the MR parse tree is similar. Given an MRL production, $A \rightarrow \beta$, where β contains non-terminals $A_1, \ldots, A_n$, a rule $A \rightarrow \langle \alpha, \lambda x_{i_1} \ldots \lambda x_{i_k}.\beta' \rangle$ is extracted such that: (1) α is the NL phrase linked to the MRL production, with non-terminals $A_1, \ldots, A_n$ showing the positions of the argument strings; (2) β' is β with each non-terminal A_j replaced with $A_j(x_{j_1}, \ldots, x_{j_{k_j}})$, where $x_{j_1}, \ldots, x_{j_{k_j}}$ are the bound variables in the λ-function used to rewrite A_j; (3) $x_{i_1}, \ldots, x_{i_k}$ are the logical variables that appear in β' and outside the current MR sub-parse. For example, see the rule

FORM → ⟨ *state* , λx_1.state(x_1) ⟩
FORM → ⟨ *by area* , $\lambda x_1.\lambda x_2$.area(x_1, x_2) ⟩
FORM → ⟨ *smallest* FORM[1] FORM[2] , λx_1.smallest$(x_2,$ (FORM[1](x_1), FORM[2](x_1, x_2))) ⟩
QUERY → ⟨ *what is* (1) FORM[1] , answer$(x_1,$ FORM[1](x_1)) ⟩

Figure 3: λ-SCFG rules for parsing the English sentence in Figure 1(a).

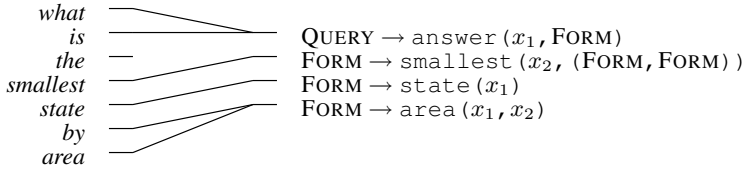


Figure 4: Word alignment for the sentence pair in Figure 1(a).

extracted for the smallest predicate in Figure 3, where x_2 is an argument of smallest, but it does not appear outside the formula smallest(...), so x_2 need not be bound by λ. On the other hand, x_1 appears in β', and it appears outside smallest(...) (as an argument of answer), so x_1 must be bound.

Rule extraction continues in this manner until the root of the MR parse tree is reached. Figure 3 shows all the rules extracted from Figure 4.[1]

3.2 Probabilistic Semantic Parsing Model

Since the learned λ-SCFG can be ambiguous, a probabilistic model is needed for parse disambiguation. We use the maximum-entropy model proposed in Wong and Mooney (2006), which defines a conditional probability distribution over derivations given an observed NL sentence. The output MR is the yield of the most probable derivation according to this model.

Parameter estimation involves maximizing the conditional log-likelihood of the training set. For each rule, r, there is a feature that returns the number of times r is used in a derivation. More features will be introduced in Section 5.

4 Promoting NL/MRL Isomorphism

We have described the λ-WASP algorithm which generates logical forms based on λ-calculus. While reasonably effective, it can be improved in several ways. In this section, we focus on improving lexical acquisition.

To see why the current lexical acquisition algorithm can be problematic, consider the word alignment in Figure 5 (for the sentence pair in Figure 1(b)). No rules can be extracted for the state predicate, because the shortest NL substring that covers the word *states* and the argument string *Texas*, i.e. *states bordering Texas*, contains the word *bordering*, which is linked to an MRL production outside the MR sub-parse rooted at state. Rule extraction is forbidden in this case because it would destroy the link between *bordering* and next_to. In other words, the NL and MR parse trees are not *isomorphic*.

This problem can be ameliorated by transforming the logical form of each training sentence so that the NL and MR parse trees are maximally isomorphic. This is possible because some of the operators used in the logical forms, notably the conjunction operator (,), are both associative (a,(b,c) = (a,b),c = a,b,c) and commutative (a,b = b,a). Hence, conjuncts can be reordered and regrouped without changing the meaning of a conjunction. For example, rule extraction would be possible if the positions of the next_to and state conjuncts were switched. We present a method for regrouping conjuncts to promote isomorphism between NL and MR parse trees.[2] Given a conjunction, it does the following: (See Figure 6 for the pseudocode, and Figure 5 for an illustration.)

Step 1. Identify the MRL productions that correspond to the conjuncts and the meta-predicate that takes the conjunction as an argument (count in Figure 5), and figure them as vertices in an undi-

[1] For details regarding non-isomorphic NL/MR parse trees, removal of bad links from alignments, and extraction of word gaps (e.g. the token (1) in the last rule of Figure 3), see Wong and Mooney (2006).

[2] This method also applies to *any* operators that are associative and commutative, e.g. disjunction. For concreteness, however, we use conjunction as an example.

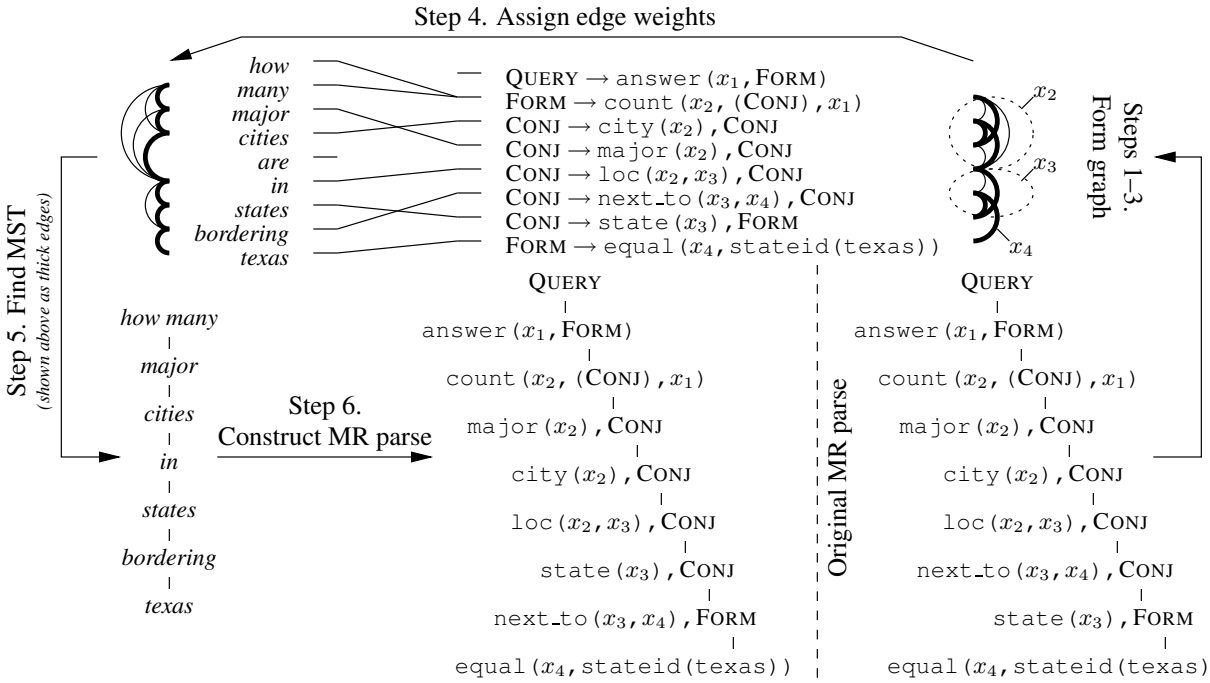


Figure 5: Transforming the logical form in Figure 1(b). The step numbers correspond to those in Figure 6.

Input: A conjunction, $\mathbf{c}$, of n conjuncts; MRL productions, $p_1, \ldots, p_n$, that correspond to each conjunct; an MRL production, p_0, that corresponds to the meta-predicate taking $\mathbf{c}$ as an argument; an NL sentence, $\mathbf{e}$; a word alignment, $\mathbf{a}$.

1 Let $\mathbf{v}(p)$ be the set of logical variables that appear in p. Create an undirected graph, Γ, with vertices $V = \{p_i | i = 0, \ldots, n\}$ and edges $E = \{(p_i, p_j) | i < j, \mathbf{v}(p_i) \cap \mathbf{v}(p_j) \neq \emptyset\}$.
2 Let $\mathbf{e}(p)$ be the set of words in $\mathbf{e}$ to which p is linked according to $\mathbf{a}$. Let $\text{span}(p_i, p_j)$ be the shortest substring of $\mathbf{e}$ that includes $\mathbf{e}(p_i) \cup \mathbf{e}(p_j)$. Subtract $\{(p_i, p_j) | i \neq 0, \text{span}(p_i, p_j) \cap \mathbf{e}(p_0) \neq \emptyset\}$ from E.
3 Add edges (p_0, p_i) to E if p_i is not already connected to p_0.
4 For each edge (p_i, p_j) in E, set edge weight to the minimum word distance between $\mathbf{e}(p_i)$ and $\mathbf{e}(p_j)$.
5 Find a minimum spanning tree, T, for Γ using Kruskal's algorithm.
6 Using p_0 as the root, construct a conjunction $\mathbf{c}'$ based on T, and then replace $\mathbf{c}$ with $\mathbf{c}'$.

Figure 6: Algorithm for regrouping conjuncts to promote isomorphism between NL and MR parse trees.

rected graph, Γ. An edge (p_i, p_j) is in Γ if and only if p_i and p_j contain occurrences of the same logical variables. Each edge in Γ indicates a possible edge in the transformed MR parse tree. Intuitively, two concepts are closely related if they involve the same logical variables, and therefore, should be placed close together in the MR parse tree. By keeping occurrences of a logical variable in close proximity in the MR parse tree, we also avoid unnecessary variable bindings in the extracted rules.

Step 2. Remove edges from Γ whose inclusion in the MR parse tree would prevent the NL and MR parse trees from being isomorphic.

Step 3. Add edges to Γ to make sure that a spanning tree for Γ exists.

Steps 4–6. Assign edge weights based on word distance, find a minimum spanning tree, T, for Γ, then regroup the conjuncts based on T. The choice of T reflects the intuition that words that occur close together in a sentence tend to be semantically related.

This procedure is repeated for all conjunctions that appear in a logical form. Rules are then extracted from the same input alignment used to regroup conjuncts. Of course, the regrouping of conjuncts requires a good alignment to begin with, and that requires a reasonable ordering of conjuncts in the training data, since the alignment model is sensitive to word order. This suggests an iterative algorithm in which a better grouping of conjuncts leads to a better alignment model, which guides further regrouping until convergence. We did not pursue this, as it is not needed in our experiments so far.

(a) answer(x_1,largest(x_2,(state(x_1),major(x_1),river(x_1),traverse(x_1,x_2))))
What is the entity that is a state and also a major river, that traverses something that is the largest?

(b) answer(x_1,smallest(x_2,(highest(x_1,(point(x_1),loc(x_1,x_3),state(x_3))),density(x_1,x_2))))
Among the highest points of all states, which one has the lowest population density?

(c) answer(x_1,equal(x_1,stateid(alaska)))
Alaska?

(d) answer(x_1,largest(x_2,(largest(x_1,(state(x_1),next_to(x_1,x_3),state(x_3))),population(x_1,x_2))))
Among the largest state that borders some other state, which is the one with the largest population?

Figure 7: Typical errors made by the λ-WASP parser, along with their English interpretations, before any language modeling for the target MRL was done.

5 Modeling the Target MRL

In this section, we propose two methods for modeling the target MRL. This is motivated by the fact that many of the errors made by the λ-WASP parser can be detected by inspecting the MR translations alone. Figure 7 shows some typical errors, which can be classified into two broad categories:

1. Type mismatch errors. For example, a state cannot possibly be a river (Figure 7(a)). Also it is awkward to talk about the population density of a state's highest point (Figure 7(b)).

2. Errors that do not involve type mismatch. For example, a query can be overly trivial (Figure 7(c)), or involve aggregate functions on a known singleton (Figure 7(d)).

The first type of errors can be fixed by type checking. Each m-place predicate is associated with a list of m-tuples showing all valid *combinations* of entity types that the m arguments can refer to:

point(_): {(POINT)}
density(_,_):
{(COUNTRY, NUM), (STATE, NUM), (CITY, NUM)}

These m-tuples of entity types are given as domain knowledge. The parser maintains a set of possible entity types for each logical variable introduced in a partial derivation (except those that are no longer visible). If there is a logical variable that cannot refer to any types of entities (i.e. the set of entity types is empty), then the partial derivation is considered invalid. For example, based on the tuples shown above, point(x_1) and density(x_1,_) cannot be both true, because $\{\text{POINT}\} \cap \{\text{COUNTRY}, \text{STATE}, \text{CITY}\} = \emptyset$. The use of type checking is to exploit the fact that people tend not to ask questions that obviously have no valid answers (Grice, 1975). It is also similar to Schuler's (2003) use of model-theoretic interpretations to guide syntactic parsing.

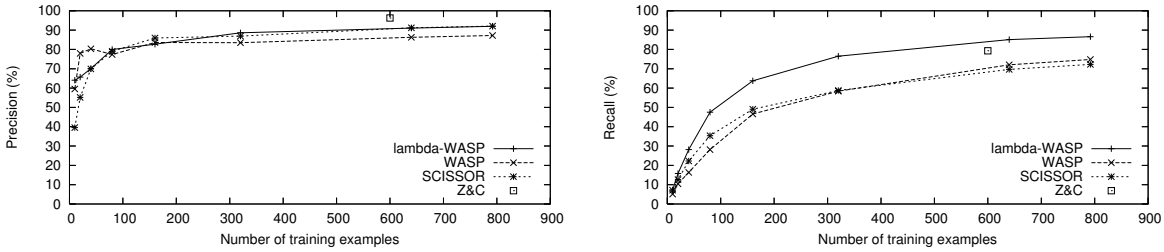
Errors that do not involve type mismatch are handled by adding new features to the maximum-entropy model (Section 3.2). We only consider features that are based on the MR translations, and therefore, these features can be seen as an implicit language model of the target MRL (Papineni et al., 1997). Of the many features that we have tried, one feature set stands out as being the most effective, the *two-level rules* in Collins and Koo (2005), which give the number of times a given rule is used to expand a non-terminal in a given parent rule. We use only the MRL part of the rules. For example, a negative weight for the combination of QUERY $\rightarrow$ answer(x_1,FORM(x_1)) and FORM $\rightarrow \lambda x_1$.equal(x_1,_) would discourage any parse that yields Figure 7(c). The two-level rules features, along with the features described in Section 3.2, are used in the final version of λ-WASP.

6 Experiments

We evaluated the λ-WASP algorithm in the GEOQUERY domain. The larger GEOQUERY corpus consists of 880 English questions gathered from various sources (Wong and Mooney, 2006). The questions were manually translated into Prolog logical forms. The average length of a sentence is 7.57 words.

We performed a single run of 10-fold cross validation, and measured the performance of the learned parsers using *precision* (percentage of translations that were correct), *recall* (percentage of test sentences that were correctly translated), and *F-measure* (harmonic mean of precision and recall). A translation is considered correct if it retrieves the same answer as the correct logical form.

Figure 8 shows the learning curves for the λ-



(a) Precision (b) Recall

Figure 8: Learning curves for various parsing algorithms on the larger GEOQUERY corpus.

(%)	λ-WASP	WASP	SCISSOR	Z&C
Precision	91.95	87.19	92.08	**96.25**
Recall	**86.59**	74.77	72.27	79.29
F-measure	**89.19**	80.50	80.98	86.95

Table 1: Performance of various parsing algorithms on the larger GEOQUERY corpus.

WASP algorithm compared to: (1) the original WASP algorithm which uses a functional query language (FunQL); (2) SCISSOR (Ge and Mooney, 2005), a fully-supervised, combined syntactic-semantic parsing algorithm which also uses FunQL; and (3) Zettlemoyer and Collins (2005) (Z&C), a CCG-based algorithm which uses Prolog logical forms. Table 1 summarizes the results at the end of the learning curves (792 training examples for λ-WASP, WASP and SCISSOR, 600 for Z&C).

A few observations can be made. First, algorithms that use Prolog logical forms as the target MRL generally show better recall than those using FunQL. In particular, λ-WASP has the best recall by far. One reason is that it allows lexical items to be combined in ways not allowed by FunQL or the hand-written templates in Z&C, e.g. [*smallest* [*state*] [*by area*]] in Figure 3. Second, Z&C has the best precision, although their results are based on 280 test examples only, whereas our results are based on 10-fold cross validation. Third, λ-WASP has the best F-measure.

To see the relative importance of each component of the λ-WASP algorithm, we performed two ablation studies. First, we compared the performance of λ-WASP with and without conjunct regrouping (Section 4). Second, we compared the performance of λ-WASP with and without language modeling for the MRL (Section 5). Table 2 shows the results. It is found that conjunct regrouping improves recall ($p < 0.01$ based on the paired t-test), and the use of *two-level rules* in the maximum-entropy model improves precision and recall ($p < 0.05$). Type checking also significantly improves precision and recall.

A major advantage of λ-WASP over SCISSOR and Z&C is that it does not require any prior knowledge of the NL syntax. Figure 9 shows the performance of λ-WASP on the multilingual GEOQUERY data set. The 250-example data set is a subset of the larger GEOQUERY corpus. All English questions in this data set were manually translated into Spanish, Japanese and Turkish, while the corresponding Prolog queries remain unchanged. Figure 9 shows that λ-WASP performed comparably for all NLs. In contrast, SCISSOR cannot be used directly on the non-English data, because syntactic annotations are only available in English. Z&C cannot be used directly either, because it requires NL-specific templates for building CCG grammars.

7 Conclusions

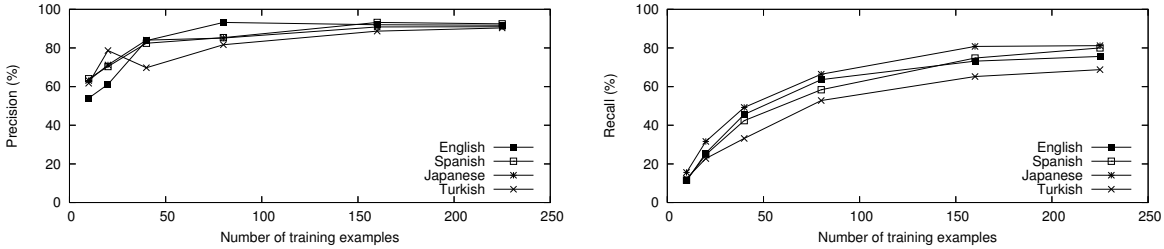
We have presented λ-WASP, a semantic parsing algorithm based on a λ-SCFG that generates logical forms using λ-calculus. A semantic parser is learned given a set of training sentences and their correct logical forms using standard SMT techniques. The result is a robust semantic parser for predicate logic, and it is the best-performing system so far in the GEOQUERY domain.

This work shows that it is possible to use standard SMT methods in tasks where logical forms are involved. For example, it should be straightforward to adapt λ-WASP to the NL generation task—all one needs is a decoder that can handle input logical forms. Other tasks that can potentially benefit from

(%)	Precision	Recall
λ-WASP	**91.95**	**86.59**
w/o conj. regrouping	**90.73**	83.07

(%)	Precision	Recall
λ-WASP	**91.95**	**86.59**
w/o two-level rules	88.46	84.32
and w/o type checking	65.45	63.18

Table 2: Performance of λ-WASP with certain components of the algorithm removed.



(a) Precision (b) Recall

Figure 9: Learning curves for λ-WASP on the multilingual GEOQUERY data set.

this include question answering and interlingual MT.

In future work, we plan to further generalize the synchronous parsing framework to allow different combinations of grammar formalisms. For example, to handle long-distance dependencies that occur in open-domain text, CCG and TAG would be more appropriate than CFG. Certain applications may require different meaning representations, e.g. frame semantics.

Acknowledgments: We thank Rohit Kate, Razvan Bunescu and the anonymous reviewers for their valuable comments. This work was supported by a gift from Google Inc.

References

A. V. Aho and J. D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*. Prentice Hall, Englewood Cliffs, NJ.

S. Bayer, J. Burger, W. Greiff, and B. Wellner. 2004. The MITRE logical form generation system. In *Proc. of Senseval-3*, Barcelona, Spain, July.

P. Blackburn and J. Bos. 2005. *Representation and Inference for Natural Language: A First Course in Computational Semantics*. CSLI Publications, Stanford, CA.

J. Bos. 2005. Towards wide-coverage semantic interpretation. In *Proc. of IWCS-05*, Tilburg, The Netherlands, January.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. of ACL-05*, pages 263–270, Ann Arbor, MI, June.

M. Collins and T. Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–69.

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 *Proc. of COLING/ACL-06*, pages 961–968, Sydney, Australia, July.

R. Ge and R. J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proc. of CoNLL-05*, pages 9–16, Ann Arbor, MI, July.

H. P. Grice. 1975. Logic and conversation. In P. Cole and J. Morgan, eds., *Syntax and Semantics 3: Speech Acts*, pages 41–58. Academic Press, New York.

A. K. Joshi and K. Vijay-Shanker. 2001. Compositional semantics with lexicalized tree-adjoining grammar (LTAG): How much underspecification is necessary? In H. Bunt et al., eds., *Computing Meaning*, volume 2, pages 147–163. Kluwer Academic Publishers, Dordrecht, The Netherlands.

R. Montague. 1970. Universal grammar. *Theoria*, 36:373–398.

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

K. A. Papineni, S. Roukos, and R. T. Ward. 1997. Feature-based language understanding. In *Proc. of EuroSpeech-97*, pages 1435–1438, Rhodes, Greece.

W. Schuler. 2003. Using model-theoretic semantic interpretation to guide statistical parsing and word recognition in a spoken language interface. In *Proc. of ACL-03*, pages 529–536.

Y. W. Wong and R. J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proc. of HLT/NAACL-06*, pages 439–446, New York City, NY.

Y. W. Wong and R. J. Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *Proc. of NAACL/HLT-07*, Rochester, NY, to appear.

D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

K. Yamada and K. Knight. 2001. A syntax-based statistical translation model. In *Proc. of ACL-01*, pages 523–530, Toulouse, France.

J. M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proc. of AAAI-96*, pages 1050–1055, Portland, OR, August.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. of UAI-05*, Edinburgh, Scotland, July.

Generalizing Tree Transformations for Inductive Dependency Parsing

Jens Nilsson[*] **Joakim Nivre**[*†] **Johan Hall**[*]

[*]Växjö University, School of Mathematics and Systems Engineering, Sweden
[†]Uppsala University, Dept. of Linguistics and Philology, Sweden
{jni,nivre,jha}@msi.vxu.se

Abstract

Previous studies in data-driven dependency parsing have shown that tree transformations can improve parsing accuracy for specific parsers and data sets. We investigate to what extent this can be generalized across languages/treebanks and parsers, focusing on pseudo-projective parsing, as a way of capturing non-projective dependencies, and transformations used to facilitate parsing of coordinate structures and verb groups. The results indicate that the beneficial effect of pseudo-projective parsing is independent of parsing strategy but sensitive to language or treebank specific properties. By contrast, the construction specific transformations appear to be more sensitive to parsing strategy but have a constant positive effect over several languages.

1 Introduction

Treebank parsers are trained on syntactically annotated sentences and a major part of their success can be attributed to extensive manipulations of the training data as well as the output of the parser, usually in the form of various tree transformations. This can be seen in state-of-the-art constituency-based parsers such as Collins (1999), Charniak (2000), and Petrov et al. (2006), and the effects of different transformations have been studied by Johnson (1998), Klein and Manning (2003), and Bikel (2004). Corresponding manipulations in the form of tree transformations for dependency-based parsers have recently gained more interest (Nivre and Nilsson, 2005; Hall and Novák, 2005; McDonald and Pereira, 2006; Nilsson et al., 2006) but are still less studied, partly because constituency-based parsing has dominated the field for a long time, and partly because dependency structures have less structure to manipulate than constituent structures.

Most of the studies in this tradition focus on a particular parsing model and a particular data set, which means that it is difficult to say whether the effect of a given transformation is dependent on a particular parsing strategy or on properties of a particular language or treebank, or both. The aim of this study is to further investigate some tree transformation techniques previously proposed for data-driven dependency parsing, with the specific aim of trying to generalize results across languages/treebanks and parsers. More precisely, we want to establish, first of all, whether the transformation as such makes specific assumptions about the language, treebank or parser and, secondly, whether the improved parsing accuracy that is due to a given transformation is constant across different languages, treebanks, and parsers.

The three types of syntactic phenomena that will be studied here are non-projectivity, coordination and verb groups, which in different ways pose problems for dependency parsers. We will focus on tree transformations that combine preprocessing with post-processing, and where the parser is treated as a black box, such as the pseudo-projective parsing technique proposed by Nivre and Nilsson (2005) and the tree transformations investigated in Nilsson et al. (2006). To study the influence of lan-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 968–975,
Prague, Czech Republic, June 2007.

guage and treebank specific properties we will use data from Arabic, Czech, Dutch, and Slovene, taken from the CoNLL-X shared task on multilingual dependency parsing (Buchholz and Marsi, 2006). To study the influence of parsing methodology, we will compare two different parsers: MaltParser (Nivre et al., 2004) and MSTParser (McDonald et al., 2005). Note that, while it is possible in principle to distinguish between syntactic properties of a language as such and properties of a particular syntactic annotation of the language in question, it will be impossible to tease these apart in the experiments reported here, since this would require having not only multiple languages but also multiple treebanks for each language. In the following, we will therefore speak about the properties of *treebanks* (rather than *languages*), but it should be understood that these properties in general depend both on properties of the language and of the particular syntactic annotation adopted in the treebank.

The rest of the paper is structured as follows. Section 2 surveys tree transformations used in dependency parsing and discusses dependencies between transformations, on the one hand, and treebanks and parsers, on the other. Section 3 introduces the four treebanks used in this study, and section 4 briefly describes the two parsers. Experimental results are presented in section 5 and conclusions in section 6.

2 Background

2.1 Non-projectivity

The tree transformations that have attracted most interest in the literature on dependency parsing are those concerned with recovering non-projectivity. The definition of non-projectivity can be found in Kahane et al. (1998). Informally, an arc is *projective* if all tokens it covers are descendants of the arc's head token, and a dependency tree is projective if all its arcs are projective.[1]

The full potential of dependency parsing can only be realized if non-projectivity is allowed, which pose a problem for projective dependency parsers. Direct non-projective parsing can be performed with good accuracy, e.g., using the Chu-Liu-Edmonds algorithm, as proposed by McDonald et al. (2005). On the other hand, non-projective parsers tend, among other things, to be slower. In order to maintain the benefits of projective parsing, tree transformations techniques to recover non-projectivity while using a projective parser have been proposed in several studies, some described below.

In discussing the recovery of empty categories in data-driven constituency parsing, Campbell (2004) distinguishes between approaches based on pure post-processing and approaches based on a combination of preprocessing and post-processing. The same division can be made for the recovery of non-projective dependencies in data-driven dependency parsing.

Pure Post-processing

Hall and Novák (2005) propose a corrective modeling approach. The motivation is that the parsers of Collins et al. (1999) and Charniak (2000) adapted to Czech are not able to create the non-projective arcs present in the treebank, which is unsatisfactory. They therefore aim to correct erroneous arcs in the parser's output (specifically all those arcs which should be non-projective) by training a classifier that predicts the most probable head of a token in the neighborhood of the head assigned by the parser.

Another example is the second-order approximate spanning tree parser developed by McDonald and Pereira (2006). It starts by producing the highest scoring projective dependency tree using Eisner's algorithm. In the second phase, tree transformations are performed, replacing lower scoring projective arcs with higher scoring non-projective ones.

Preprocessing with Post-processing

The training data can also be preprocessed to facilitate the recovery of non-projective arcs in the output of a projective parser. The pseudo-projective transformation proposed by Nivre and Nilsson (2005) is such an approach, which is compatible with different parser engines.

First, the training data is *projectivized* by making non-projective arcs projective using a lifting operation. This is combined with an augmentation of the dependency labels of projectivized arcs (and/or surrounding arcs) with information that probably reveals their correct non-projective positions. The out-

[1] If dependency arcs are drawn above the linearly ordered sequence of tokens, preceded by a special root node, then a non-projective dependency tree always has crossing arcs.

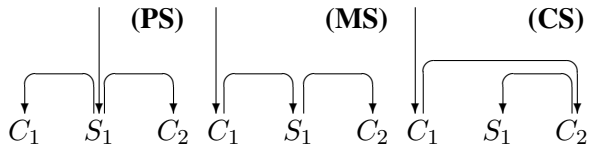


Figure 1: Dependency structure for coordination

put of the parser, trained on the projectivized data, is then *deprojectivized* by a heuristic search using the added information in the dependency labels. The only assumption made about the parser is therefore that it can learn to derive labeled dependency structures with augmented dependency labels.

2.2 Coordination and Verb Groups

The second type of transformation concerns linguistic phenomena that are not impossible for a projective parser to process but which may be difficult to learn, given a certain choice of dependency analysis. This study is concerned with two such phenomena, coordination and verb groups, for which tree transformations have been shown to improve parsing accuracy for MaltParser on Czech (Nilsson et al., 2006). The general conclusion of this study is that coordination and verb groups in the Prague Dependency Treebank (PDT), based on theories of the Prague school (PS), are annotated in a way that is difficult for the parser to learn. By transforming coordination and verb groups in the training data to an annotation similar to that advocated by Mel'čuk (1988) and then performing an inverse transformation on the parser output, parsing accuracy can therefore be improved. This is again an instance of the black-box idea.

Schematically, coordination is annotated in the Prague school as depicted in PS in figure 1, where the conjuncts are dependents of the conjunction. In Mel'čuk style (MS), on the other hand, conjuncts and conjunction(s) form a chain going from left to right. A third way of treating coordination, not discussed by Nilsson et al. (2006), is used by the parser of Collins (1999), which internally represents coordination as a direct relation between the conjuncts. This is illustrated in CS in figure 1, where the conjunction depends on one of the conjuncts, in this case on the rightmost one.

Nilsson et al. (2006) also show that the annotation of verb groups is not well-suited for parsing PDT using MaltParser, and that transforming the dependency structure for verb groups has a positive impact on parsing accuracy. In PDT, auxiliary verbs are dependents of the main verb, whereas it according to Mel'čuk is the (finite) auxiliary verb that is the head of the main verb. Again, the parsing experiments in this study show that verb groups are more difficult to parse in PS than in MS.

2.3 Transformations, Parsers, and Treebanks

Pseudo-projective parsing and transformations for coordination and verb groups are instances of the same general methodology:

1. Apply a tree transformation to the training data.
2. Train a parser on the transformed data.
3. Parse new sentences.
4. Apply an inverse transformation to the output of the parser.

In this scheme, the parser is treated as a black box. All that is assumed is that it is a data-driven parser designed for (projective) labeled dependency structures. In this sense, the tree transformations are independent of parsing methodology. Whether the beneficial effect of a transformation, if any, is also independent of parsing methodology is another question, which will be addressed in the experimental part of this paper.

The pseudo-projective transformation is independent not only of parsing methodology but also of treebank (and language) specific properties, as long as the target representation is a (potentially non-projective) labeled dependency structure. By contrast, the coordination and verb group transformations presuppose not only that the language in question contains these constructions but also that the treebank adopts a PS annotation. In this sense, they are more limited in their applicability than pseudo-projective parsing. Again, it is a different question whether the transformations have a positive effect for all treebanks (languages) to which they can be applied.

3 Treebanks

The experiments are mostly conducted using treebank data from the CoNLL shared task 2006. This

	Slovene SDT	Arabic PADT	Dutch Alpino	Czech PDT
# T	29	54	195	1249
# S	1.5	1.5	13.3	72.7
%-NPS	22.2	11.2	36.4	23.2
%-NPA	1.8	0.4	5.4	1.9
%-C	9.3	8.5	4.0	8.5
%-A	8.8	-	-	1.3

Table 1: Overview of the data sets (ordered by size), where # S * 1000 = number of sentences, # T * 1000 = number of tokens, %-NPS = percentage of non-projective sentences, %-NPA = percentage of non-projective arcs, %-C = percentage of conjuncts, %-A = percentage of auxiliary verbs.

subsection summarizes some of the important characteristics of these data sets, with an overview in table 1. Any details concerning the conversion from the original formats of the various treebanks to the CoNLL format, a pure dependency based format, are found in documentation referred to in Buchholz and Marsi (2006).

PDT (Hajič et al., 2001) is the largest manually annotated treebank, and as already mentioned, it adopts PS for coordination and verb groups. As the last four rows reveal, PDT contains a quite high proportion of non-projectivity, since almost every fourth dependency graph contains at least one non-projective arc. The table also shows that coordination is more common than verb groups in PDT. Only 1.3% of the tokens in the training data are identified as auxiliary verbs, whereas 8.5% of the tokens are identified as conjuncts.

Both Slovene Dependency Treebank (Džeroski et al., 2006) (SDT) and Prague Arabic Dependency Treebank (Hajič et al., 2004) (PADT) annotate coordination and verb groups as in PDT, since they too are influenced by the theories of the Prague school. The proportions of non-projectivity and conjuncts in SDT are in fact quite similar to the proportions in PDT. The big difference is the proportion of auxiliary verbs, with many more auxiliary verbs in SDT than in PDT. It is therefore plausible that the transformations for verb groups will have a larger impact on parser accuracy in SDT.

Arabic is not a Slavic languages such as Czech and Slovene, and the annotation in PADT is therefore more dissimilar to PDT than SDT is. One such example is that Arabic does not have auxiliary verbs. Table 1 thus does not give figures verb groups. The amount of coordination is on the other hand comparable to both PDT and SDT. The table also reveals that the amount of non-projective arcs is about 25% of that in PDT and SDT, although the amount of non-projective sentences is still as large as 50% of that in PDT and SDT.

Alpino (van der Beek et al., 2002) in the CoNLL format, the second largest treebank in this study, is not as closely tied to the theories of the Prague school as the others, but still treats coordination in a way similar to PS. The table shows that coordination is less frequent in the CoNLL version of Alpino than in the three other treebanks. The other characteristic of Alpino is the high share of non-projectivity, where more than every third sentence is non-projective. Finally, the lack of information about the share of auxiliary verbs is not due to the non-existence of such verbs in Dutch but to the fact that Alpino adopts an MS annotation of verb groups (i.e., treating main verbs as dependents of auxiliary verbs), which means that the verb group transformation of Nilsson et al. (2006) is not applicable.

4 Parsers

The parsers used in the experiments are MaltParser (Nivre et al., 2004) and MSTParser (McDonald et al., 2005). These parsers are based on very different parsing strategies, which makes them suitable in order to test the parser independence of different transformations. MaltParser adopts a greedy, deterministic parsing strategy, deriving a labeled dependency structure in a single left-to-right pass over the input and uses support vector machines to predict the next parsing action. MSTParser instead extracts a maximum spanning tree from a dense weighted graph containing all possible dependency arcs between tokens (with Eisner's algorithm for projective dependency structures or the Chu-Liu-Edmonds algorithm for non-projective structures), using a global discriminative model and online learning to assign weights to individual arcs.[2]

[2]The experiments in this paper are based on the first-order factorization described in McDonald et al. (2005)

5 Experiments

The experiments reported in section 5.1–5.2 below are based on the training sets from the CoNLL-X shared task, except where noted. The results reported are obtained by a ten-fold cross-validation (with a pseudo-randomized split) for all treebanks except PDT, where 80% of the data was used for training and 20% for development testing (again with a pseudo-randomized split). In section 5.3, we give results for the final evaluation on the CoNLL-X test sets using all three transformations together with MaltParser.

Parsing accuracy is primarily measured by the unlabeled attachment score (AS_U), i.e., the proportion of tokens that are assigned the correct head, as computed by the official CoNLL-X evaluation script with default settings (thus excluding all punctuation tokens). In section 5.3 we also include the *labeled* attachment score (AS_L) (where a token must have both the correct head and the correct dependency label to be counted as correct), which was the official evaluation metric in the CoNLL-X shared task.

5.1 Comparing Treebanks

We start by examining the effect of transformations on data from different treebanks (languages), using a single parser: MaltParser.

Non-projectivity

The question in focus here is whether the effect of the pseudo-projective transformation for MaltParser varies with the treebank. Table 2 presents the unlabeled attachment score results (AS_U), comparing the pseudo-projective parsing technique (P-Proj) with two baselines, obtained by training the strictly projective parser on the original (non-projective) training data (N-Proj) and on projectivized training data with no augmentation of dependency labels (Proj).

The first thing to note is that pseudo-projective parsing gives a significant improvement for PDT, as previously reported by Nivre and Nilsson (2005), but also for Alpino, where the improvement is even larger, presumably because of the higher proportion of non-projective dependencies in the Dutch treebank. By contrast, there is no significant improvement for either SDT or PADT, and even a small drop in the accuracy figures for SDT. Finally, in contrast to the results reported by Nivre and Nilsson (2005), simply projectivizing the training data (without using an inverse transformation) is not beneficial at all, except possibly for Alpino.

	N-Proj	**Proj**	**P-Proj**
SDT	**77.27**	76.63**	77.11
PADT	76.96	**77.07***	**77.07***
Alpino	82.75	83.28**	**87.08****
PDT	83.41	83.32**	**84.42****

Table 2: AS_U for pseudo-projective parsing with MaltParser. McNemar's test: $^{*} = p < .05$ and $^{**} = p < 0.01$ compared to **N-Proj**.

	1	**2**	**3**	**>3**
SDT	88.4	9.1	1.7	0.84
PADT	66.5	14.4	5.2	13.9
Alpino	84.6	13.8	1.5	0.07
PDT	93.8	5.6	0.5	0.1

Table 3: The number of lifts for non-projective arcs.

But why does not pseudo-projective parsing improve accuracy for SDT and PADT? One possible factor is the complexity of the non-projective constructions, which can be measured by counting the number of lifts that are required to make non-projective arcs projective. The more deeply nested a non-projective arc is, the more difficult it is to recover because of parsing errors as well as search errors in the inverse transformation. The figures in table 3 shed some interesting light on this factor.

For example, whereas 93.8% of all arcs in PDT require only one lift before they become projective (88.4% and 84.6% for SDT and Alpino, respectively), the corresponding figure for PADT is as low as 66.5%. PADT also has a high proportion of very deeply nested non-projective arcs (>3) in comparison to the other treebanks, making the inverse transformation for PADT more problematic than for the other treebanks. The absence of a positive effect for PADT is therefore understandable given the deeply nested non-projective constructions in PADT.

However, one question that still remains is why SDT and PDT, which are so similar in terms of both nesting depth and amount of non-projectivity, be-

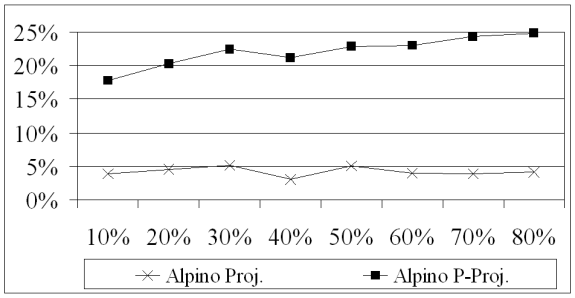


Figure 2: Learning curves for Alpino measured as error reduction for AS_U.

	None	Coord	VG
SDT	77.27	79.33**	77.92**
PADT	76.96	79.05**	-
Alpino	82.75	83.38**	-
PDT	83.41	85.51**	83.58**

Table 4: AS_U for coordination and verb group transformations with MaltParser (None = N-Proj). McNemar's test: ** $= p < .01$ compared to **None**.

have differently with respect to pseudo-projective parsing. Another factor that may be important here is the amount of training data available. As shown in table 1, PDT is more than 40 times larger than SDT. To investigate the influence of training set size, a learning curve experiment has been performed. Alpino is a suitable data set for this due to its relatively large amount of both data and non-projectivity.

Figure 2 shows the learning curve for pseudo-projective parsing (P-Proj), compared to using only projectivized training data (Proj), measured as error reduction in relation to the original non-projective training data (N-Proj). The experiment was performed by incrementally adding cross-validation folds 1–8 to the training set, using folds 9–0 as static test data.

One can note that the error reduction for Proj is unaffected by the amount of data. While the error reduction varies slightly, it turns out that the error reduction is virtually the same for 10% of the training data as for 80%. That is, there is no correlation if information concerning the lifts are not added to the labels. However, with a pseudo-projective transformation, which actively tries to recover non-projectivity, the learning curve clearly indicates that the amount of data matters. Alpino, with 36% non-projective sentences, starts at about 17% and has a climbing curve up to almost 25%.

Although this experiment shows that there is a correlation between the amount of data and the accuracy for pseudo-projective parsing, it does probably not tell the whole story. If it did, one would expect that the error reduction for the pseudo-projective transformation would be much closer to Proj when the amount of data is low (to the left in the figure) than they apparently are. Of course, the difference is likely to diminish with even less data, but it should be noted that 10% of Alpino has about half the size of PADT, for which the positive impact of pseudo-projective parsing is absent. The absence of increased accuracy for SDT can partially be explained by the higher share of non-projective arcs in Alpino (3 times more).

Coordination and Verb Groups

The corresponding parsing results using MaltParser with transformations for coordination and verb groups are shown in table 4. For SDT, PADT and PDT, the annotation of coordination has been transformed from PS to MS, as described in Nilsson et al. (2006). For Alpino, the transformation is from PS to CS (cf. section 2.2), which was found to give slightly better performance in preliminary experiments. The baseline with no transformation (None) is the same as N-Proj in table 2.

As the figures indicate, transforming coordination is beneficial not only for PDT, as reported by Nilsson et al. (2006), but also for SDT, PADT, and Alpino. It is interesting to note that SDT, PADT and PDT, with comparable amounts of conjuncts, have comparable increases in accuracy (about 2 percentage points each), despite the large differences in training set size. It is therefore not surprising that Alpino, with a much smaller amount of conjuncts, has a lower increase in accuracy. Taken together, these results indicate that the frequency of the construction is more important than the size of the training set for this type of transformation.

The same generalization over treebanks holds for verb groups too. The last column in table 4 shows that the expected increase in accuracy for PDT is ac-

Algorithm	N-Proj	Proj	P-Proj
Eisner	81.79	83.23	**86.45**
CLE	86.39		

Table 5: Pseudo-projective parsing results (AS_U) for Alpino with MSTParser.

companied by a even higher increase for SDT. This can probably be attributed to the higher frequency of auxiliary verbs in SDT.

5.2 Comparing Parsers

The main question in this section is to what extent the positive effect of different tree transformations is dependent on parsing strategy, since all previous experiments have been performed with a single parser (MaltParser). For comparison we have performed two experiments with MSTParser, version 0.1, which is based on a very different parsing methdology (cf. section 4). Due to some technical difficulties (notably the very high memory consumption when using MSTParser for labeled dependency parsing), we have not been able to replicate the experiments from the preceding section exactly. The results presented below must therefore be regarded as a preliminary exploration of the dependencies between tree transformations and parsing strategy.

Table 5 presents AS_U results for MSTParser in combination with pseudo-projective parsing applied to the Alpino treebank of Dutch.[3] The first row contains the result for Eisner's algorithm using no transformation (N-Proj), projectivized training data (Proj), and pseudo-projective parsing (P-Proj). The figures show a pattern very similar to that for MaltParser, with a boost in accuracy for Proj compared to N-Proj, and with a significantly higher accuracy for P-Proj over Proj. It is also worth noting that the error reduction between N-Proj and P-Proj is actually higher for MSTParser here than for MaltParser in table 2.

The second row contains the result for the Chu-Liu-Edmonds algorithm (CLE), which constructs non-projective structures directly and therefore does not require the pseudo-projective transformation. A comparison between Eisner's algorithm with pseudo-projective transformation and CLE reveals that pseudo-projective parsing is at least as accurate as non-projective parsing for AS_U. (The small difference is not statistically significant.)

[3]The figures are not completely comparable to the previously presented Dutch results for MaltParser, since MaltParser's feature model has access to all the information in the CoNLL data format, whereas MSTParser in this experiment only could handle word forms and part-of-speech tags.

Trans.	None	Coord	VG
AS_U	84.5	83.5	84.5

Table 6: Coordination and verb group transformations for PDT with the CLE algorithm.

		Dev	Eval	Niv	McD
SDT	AS_U	80.40	82.01	78.72	83.17
	AS_L	71.06	72.44	70.30	73.44
PADT	AS_U	78.97	78.56	77.52	79.34
	AS_L	67.63	67.58	66.71	66.91
Alpino	AS_U	87.63	82.85	81.35	83.57
	AS_L	84.02	79.73	78.59	79.19
PDT	AS_U	85.72	85.98	84.80	87.30
	AS_L	78.56	78.80	78.42	80.18

Table 7: Evaluation on CoNLL-X test data; MaltParser with all transformations (**Dev** = development, **Eval** = CoNLL test set, **Niv** = Nivre et al. (2006), **McD** = McDonald et al. (2006))

By contrast, no positive effect could be detected for the coordination and verb group transformations togther with MSTParser. The figures in table 6 are not based on CoNLL data, but instead on the evaluation test set of the original PDT 1.0, which enables a direct comparison to McDonald et. al. (2005) (the **None** column). We see that there is even a negative effect for the coordination transformation. These results clearly indicate that the effect of these transformations is at least partly dependent on parsing strategy, in contrast to what was found for the pseudo-projective parsing technique.

5.3 Combining Transformations

In order to assess the combined effect of all three transformations in relation to the state of the art, we performed a final evaluation using MaltParser on the dedicated test sets from the CoNLL-X shared task. Table 7 gives the results for both development (cross-validation for SDT, PADT, and Alpino;

development set for PDT) and final test, compared to the two top performing systems in the shared task, MSTParser with approximate second-order non-projective parsing (McDonald et al., 2006) and MaltParser with pseudo-projective parsing (but no coordination or verb group transformations) (Nivre et al., 2006). Looking at the labeled attachment score (AS_L), the official scoring metric of the CoNLL-X shared task, we see that the combined effect of the three transformations boosts the performance of MaltParser for all treebanks and in two cases out of four outperforms MSTParser (which was the top scoring system for all four treebanks).

6 Conclusion

In this paper, we have examined the generality of tree transformations for data-driven dependency parsing. The results indicate that the pseudo-projective parsing technique has a positive effect on parsing accuracy that is independent of parsing methodology but sensitive to the amount of training data as well as to the complexity of non-projective constructions. By contrast, the construction-specific transformations targeting coordination and verb groups appear to have a more language-independent effect (for languages to which they are applicable) but do not help for all parsers. More research is needed in order to know exactly what the dependencies are between parsing strategy and tree transformations. Regardless of this, however, it is safe to conclude that pre-processing and post-processing is important not only in constituency-based parsing, as previously shown in a number of studies, but also for inductive dependency parsing.

References

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

S. Buchholz and E. Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proceedings of CoNLL*, pages 1–17.

R. Campbell. 2004. Using Linguistic Principles to Recover Empty Categories. In *Proceedings of ACL*, pages 645–652.

E. Charniak. 2000. A Maximum-Entropy-Inspired Parser. In *Proceedings of NAACL*, pages 132–139.

M. Collins, J. Hajič, L. Ramshaw, and C. Tillmann. 1999. A statistical parser for Czech. In *Proceedings of ACL*, pages 100–110.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

S. Džeroski, T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtsky, and A. Žele. 2006. Towards a Slovene Dependency Treebank. In *LREC*.

J. Hajič, B. V. Hladka, J. Panevová, Eva Hajičová, Petr Sgall, and Petr Pajas. 2001. Prague Dependency Treebank 1.0. LDC, 2001T10.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic Dependency Treebank: Development in Data and Tools. In *NEMLAR*, pages 110–117.

K. Hall and V. Novák. 2005. Corrective modeling for non-projective dependency parsing. In *Proceedings of IWPT*, pages 42–52.

M. Johnson. 1998. PCFG Models of Linguistic Tree Representations. *Computational Linguistics*, 24:613–632.

S. Kahane, A. Nasr, and O. Rambow. 1998. Pseudo-Projectivity: A Polynomially Parsable Non-Projective Dependency Grammar. In *Proceedings of COLING/ACL*, pages 646–652.

D. Klein and C. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL*, pages 423–430.

R. McDonald and F. Pereira. 2006. Online Learning of Approximate Dependency Parsing Algorithms. In *Proceedings of EACL*, pages 81–88.

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

R. McDonald, K. Lerman, and F. Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of CoNLL*, pages 216–220.

I. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press.

J. Nilsson, J. Nivre, and J. Hall. 2006. Graph Transformations in Data-Driven Dependency Parsing. In *Proceedings of COLING/ACL*, pages 257–264.

J. Nivre and J. Nilsson. 2005. Pseudo-Projective Dependency Parsing. In *Proceedings of ACL*, pages 99–106.

J. Nivre, J. Hall, and J. Nilsson. 2004. Memory-based Dependency Parsing. In H. T. Ng and E. Riloff, editors, *Proceedings of CoNLL*, pages 49–56.

J. Nivre, J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov. 2006. Labeled Pseudo-Projective Dependency Parsing with Support Vector Machines. In *Proceedings of CoNLL*, pages 221–225.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of COLING/ACL*, pages 433–440.

L. van der Beek, G. Bouma, R. Malouf, and G. van Noord. 2002. The Alpino dependency treebank. In *Computational Linguistics in the Netherlands (CLIN)*.

Learning Multilingual Subjective Language via Cross-Lingual Projections

Rada Mihalcea and **Carmen Banea**
Department of Computer Science
University of North Texas
rada@cs.unt.edu, carmenb@unt.edu

Janyce Wiebe
Department of Computer Science
University of Pittsburgh
wiebe@cs.pitt.edu

Abstract

This paper explores methods for generating subjectivity analysis resources in a new language by leveraging on the tools and resources available in English. Given a bridge between English and the selected target language (e.g., a bilingual dictionary or a parallel corpus), the methods can be used to rapidly create tools for subjectivity analysis in the new language.

1 Introduction

There is growing interest in the automatic extraction of opinions, emotions, and sentiments in text (*subjectivity*), to provide tools and support for various natural language processing applications. Most of the research to date has focused on English, which is mainly explained by the availability of resources for subjectivity analysis, such as lexicons and manually labeled corpora.

In this paper, we investigate methods to automatically generate resources for subjectivity analysis for a new target language by leveraging on the resources and tools available for English, which in many cases took years of work to complete. Specifically, through experiments with cross-lingual projection of subjectivity, we seek answers to the following questions.

First, can we derive a subjectivity lexicon for a new language using an existing English subjectivity lexicon and a bilingual dictionary? Second, can we derive subjectivity-annotated corpora in a new language using existing subjectivity analysis tools for English and a parallel corpus? Finally, third, can we build tools for subjectivity analysis for a new target language by relying on these automatically generated resources?

We focus our experiments on Romanian, selected as a representative of the large number of languages that have only limited text processing resources developed to date. Note that, although we work with Romanian, the methods described are applicable to any other language, as in these experiments we (purposely) do not use any language-specific knowledge of the target language. Given a bridge between English and the selected target language (e.g., a bilingual dictionary or a parallel corpus), the methods can be applied to other languages as well.

After providing motivations, we present two approaches to developing sentence-level subjectivity classifiers for a new target language. The first uses a subjectivity lexicon translated from an English one. The second uses an English subjectivity classifier and a parallel corpus to create target-language training data for developing a statistical classifier.

2 Motivation

Automatic subjectivity analysis methods have been used in a wide variety of text processing applications, such as tracking sentiment timelines in online forums and news (Lloyd et al., 2005; Balog et al., 2006), review classification (Turney, 2002; Pang et al., 2002), mining opinions from product reviews (Hu and Liu, 2004), automatic expressive text-to-speech synthesis (Alm et al., 2005), text semantic analysis (Wiebe and Mihalcea, 2006; Esuli and Sebastiani, 2006), and question answering (Yu and Hatzivassiloglou, 2003).

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 976–983,
Prague, Czech Republic, June 2007.

While much recent work in subjectivity analysis focuses on *sentiment* (a type of subjectivity, namely positive and negative emotions, evaluations, and judgments), we opt to focus on recognizing subjectivity in general, for two reasons.

First, even when sentiment is the desired focus, researchers in sentiment analysis have shown that a two-stage approach is often beneficial, in which subjective instances are distinguished from objective ones, and then the subjective instances are further classified according to polarity (Yu and Hatzivassiloglou, 2003; Pang and Lee, 2004; Wilson et al., 2005; Kim and Hovy, 2006). In fact, the problem of distinguishing subjective versus objective instances has often proved to be more difficult than subsequent polarity classification, so improvements in subjectivity classification promise to positively impact sentiment classification. This is reported in studies of manual annotation of phrases (Takamura et al., 2006), recognizing contextual polarity of expressions (Wilson et al., 2005), and sentiment tagging of words and word senses (Andreevskaia and Bergler, 2006; Esuli and Sebastiani, 2006).

Second, an NLP application may seek a wide range of types of subjectivity attributed to a person, such as their motivations, thoughts, and speculations, in addition to their positive and negative sentiments. For instance, the opinion tracking system Lydia (Lloyd et al., 2005) gives separate ratings for subjectivity and sentiment. These can be detected with subjectivity analysis but not by a method focused only on sentiment.

There is world-wide interest in text analysis applications. While work on subjectivity analysis in other languages is growing (e.g., Japanese data are used in (Takamura et al., 2006; Kanayama and Nasukawa, 2006), Chinese data are used in (Hu et al., 2005), and German data are used in (Kim and Hovy, 2006)), much of the work in subjectivity analysis has been applied to English data. Creating corpora and lexical resources for a new language is very time consuming. In general, we would like to leverage resources already developed for one language to more rapidly create subjectivity analysis tools for a new one. This motivates our exploration and use of cross-lingual lexicon translations and annotation projections.

Most if not all work on subjectivity analysis has been carried out in a monolingual framework. We are not aware of multi-lingual work in subjectivity analysis such as that proposed here, in which subjectivity analysis resources developed for one language are used to support developing resources in another.

3 A Lexicon-Based Approach

Many subjectivity and sentiment analysis tools rely on manually or semi-automatically constructed lexicons (Yu and Hatzivassiloglou, 2003; Riloff and Wiebe, 2003; Kim and Hovy, 2006). Given the success of such techniques, the first approach we take to generating a target-language subjectivity classifier is to create a subjectivity lexicon by translating an existing source language lexicon, and then build a classifier that relies on the resulting lexicon.

Below, we describe the translation process and discuss the results of an annotation study to assess the quality of the translated lexicon. We then describe and evaluate a lexicon-based target-language classifier.

3.1 Translating a Subjectivity Lexicon

The subjectivity lexicon we use is from OpinionFinder (Wiebe and Riloff, 2005), an English subjectivity analysis system which, among other things, classifies sentences as subjective or objective. The lexicon was compiled from manually developed resources augmented with entries learned from corpora. It contains 6,856 unique entries, out of which 990 are multi-word expressions. The entries in the lexicon have been labeled for part of speech, and for reliability – those that appear most often in subjective contexts are *strong* clues of subjectivity, while those that appear less often, but still more often than expected by chance, are labeled *weak*.

To perform the translation, we use two bilingual dictionaries. The first is an authoritative English-Romanian dictionary, consisting of 41,500 entries,[1] which we use as the main translation resource for the lexicon translation. The second dictionary, drawn from the Universal Dictionary download site (UDP, 2007) consists of 4,500 entries written largely by Web volunteer contributors, and thus is not error free. We use this dictionary only for those entries that do not appear in the main dictionary.

[1]Unique English entries, each with multiple Romanian translations.

There were several challenges encountered in the translation process. First, although the English subjectivity lexicon contains inflected words, we must use the lemmatized form in order to be able to translate the entries using the bilingual dictionary. However, words may lose their subjective meaning once lemmatized. For instance, the inflected form of *memories* becomes *memory*. Once translated into Romanian (as *memorie*), its main meaning is objective, referring to the power of retaining information as in *Iron supplements may improve a woman's memory*.

Second, neither the lexicon nor the bilingual dictionary provides information on the sense of the individual entries, and therefore the translation has to rely on the most probable sense in the target language. Fortunately, the bilingual dictionary lists the translations in reverse order of their usage frequencies. Nonetheless, the ambiguity of the words and the translations still seems to represent an important source of error. Moreover, the lexicon sometimes includes identical entries expressed through different parts of speech, e.g., *grudge* has two separate entries, for its noun and verb roles, respectively. On the other hand, the bilingual dictionary does not make this distinction, and therefore we have again to rely on the "most frequent" heuristic captured by the translation order in the bilingual dictionary.

Finally, the lexicon includes a significant number (990) of multi-word expressions that pose translation difficulties, sometimes because their meaning is idiomatic, and sometimes because the multi-word expression is not listed in the bilingual dictionary and the translation of the entire phrase is difficult to reconstruct from the translations of the individual words. To address this problem, when a translation is not found in the dictionary, we create one using a word-by-word approach. These translations are then validated by enforcing that they occur at least three times on the Web, using counts collected from the AltaVista search engine. The multi-word expressions that are not validated in this process are discarded, reducing the number of expressions from an initial set of 990 to a final set of 264.

The final subjectivity lexicon in Romanian contains 4,983 entries. Table 1 shows examples of entries in the Romanian lexicon, together with their corresponding original English form. The table also shows the reliability of the expression (*weak* or *strong*) and the part of speech – attributes that are provided in the English subjectivity lexicon.

Romanian	English	attributes
înfrumuseţa	beautifying	strong, verb
notabil	notable	weak, adj
plin de regret	full of regrets	strong, adj
sclav	slaves	weak, noun

Table 1: Examples of entries in the Romanian subjectivity lexicon

Manual Evaluation.
We want to assess the quality of the translated lexicon, and compare it to the quality of the original English lexicon. The English subjectivity lexicon was evaluated in (Wiebe and Riloff, 2005) against a corpus of English-language news articles manually annotated for subjectivity (the *MPQA* corpus (Wiebe et al., 2005)). According to this evaluation, 85% of the instances of the clues marked as *strong* and 71.5% of the clues marked as *weak* are in subjective sentences in the MPQA corpus.

Since there is no comparable Romanian corpus, an alternate way to judge the subjectivity of a Romanian lexicon entry is needed.

Two native speakers of Romanian annotated the subjectivity of 150 randomly selected entries. Each annotator independently read approximately 100 examples of each drawn from the Web, including a large number from news sources. The subjectivity of a word was consequently judged in the contexts where it most frequently appears, accounting for its most frequent meanings on the Web.

The tagset used for the annotations consists of *S(ubjective)*, *O(bjective)*, and *B(oth)*. A *W(rong)* label is also used to indicate a wrong translation. Table 2 shows the contingency table for the two annotators' judgments on this data.

	S	O	B	W	Total
S	53	6	9	0	68
O	1	27	1	0	29
B	5	3	18	0	26
W	0	0	0	27	27
Total	59	36	28	27	150

Table 2: Agreement on 150 entries in the Romanian lexicon

Without counting the wrong translations, the agreement is measured at 0.80, with a Kappa κ =

0.70, which indicates consistent agreement. After the disagreements were reconciled through discussions, the final set of 123 correctly translated entries does include 49.6% (61) subjective entries, but fully 23.6% (29) were found in the study to have primarily objective uses (the other 26.8% are mixed).

Thus, this study suggests that the Romanian subjectivity clues derived through translation are less reliable than the original set of English clues. In several cases, the subjectivity is lost in the translation, mainly due to word ambiguity in either the source or target language, or both. For instance, the word *fragile* correctly translates into Romanian as *fragil*, yet this word is frequently used to refer to breakable objects, and it loses its subjective meaning of *delicate*. Other words, such as *one-sided*, completely lose subjectivity once translated, as it becomes in Romanian *cu o singura latură*, meaning *with only one side* (as of objects).

Interestingly, the reliability of clues in the English lexicon seems to help preserve subjectivity. Out of the 77 entries marked as *strong*, 11 were judged to be objective in Romanian (14.3%), compared to 14 objective Romanian entries obtained from the 36 *weak* English clues (39.0%).

3.2 Rule-based Subjectivity Classifier Using a Subjectivity Lexicon

Starting with the Romanian lexicon, we developed a lexical classifier similar to the one introduced by (Riloff and Wiebe, 2003). At the core of this method is a high-precision subjectivity and objectivity classifier that can label large amounts of raw text using only a subjectivity lexicon. Their method is further improved with a bootstrapping process that learns extraction patterns. In our experiments, however, we apply only the rule-based classification step, since the extraction step cannot be implemented without tools for syntactic parsing and information extraction not available in Romanian.

The classifier relies on three main heuristics to label subjective and objective sentences: (1) if two or more strong subjective expressions occur in the same sentence, the sentence is labeled *Subjective*; (2) if no strong subjective expressions occur in a sentence, and at most two weak subjective expressions occur in the previous, current, and next sentence combined, then the sentence is labeled *Objective*; (3) otherwise, if none of the previous rules apply, the sentence is labeled *Unknown*.

The quality of the classifier was evaluated on a Romanian gold-standard corpus annotated for subjectivity. Two native Romanian speakers (Ro_1 and Ro_2) manually annotated the subjectivity of the sentences of five randomly selected documents (504 sentences) from the Romanian side of an English-Romanian parallel corpus, according to the annotation scheme in (Wiebe et al., 2005). Agreement between annotators was measured, and then their differences were adjudicated. The baseline on this data set is 54.16%, which can be obtained by assigning a default *Subjective* label to all sentences. (More information about the corpus and annotations are given in Section 4 below, where agreement between English and Romanian aligned sentences is also assessed.)

As mentioned earlier, due to the lexicon projection process that is performed via a bilingual dictionary, the entries in our Romanian subjectivity lexicon are in a lemmatized form. Consequently, we also lemmatize the gold-standard corpus, to allow for the identification of matches with the lexicon. For this purpose, we use the Romanian lemmatizer developed by Ion and Tufiş (Ion, 2007), which has an estimated accuracy of 98%.[2]

Table 3 shows the results of the rule-based classifier. We show the precision, recall, and F-measure independently measured for the subjective, objective, and all sentences. We also evaluated a variation of the rule-based classifier that labels a sentence as objective if there are at most three weak expressions in the previous, current, and next sentence combined, which raises the recall of the objective classifier. Our attempts to increase the recall of the subjective classifier all resulted in significant loss in precision, and thus we kept the original heuristic.

In its original English implementation, this system was proposed as being high-precision but low coverage. Evaluated on the MPQA corpus, it has subjective precision of 90.4, subjective recall of 34.2, objective precision of 82.4, and objective recall of 30.7; overall, precision is 86.7 and recall is 32.6 (Wiebe and Riloff, 2005). We see a similar behavior on Romanian for subjective sentences. The subjective precision is good, albeit at the cost of low

[2] Dan Tufiş, personal communication.

Measure	Subjective	Objective	All
subj = at least two strong; obj = at most two weak			
Precision	80.00	56.50	62.59
Recall	20.51	48.91	33.53
F-measure	32.64	52.52	43.66
subj = at least two strong; obj = at most three weak			
Precision	80.00	56.85	61.94
Recall	20.51	61.03	39.08
F-measure	32.64	58.86	47.93

Table 3: Evaluation of the rule-based classifier

recall, and thus the classifier could be used to harvest subjective sentences from unlabeled Romanian data (e.g., for a subsequent bootstrapping process). The system is not very effective for objective classification, however. Recall that the objective classifier relies on the weak subjectivity clues, for which the transfer of subjectivity in the translation process was particularly low.

4 A Corpus-Based Approach

Given the low number of subjective entries found in the automatically generated lexicon and the subsequent low recall of the lexical classifier, we decided to also explore a second, corpus-based approach. This approach builds a subjectivity-annotated corpus for the target language through projection, and then trains a statistical classifier on the resulting corpus (numerous statistical classifiers have been trained for subjectivity or sentiment classification, e.g., (Pang et al., 2002; Yu and Hatzivassiloglou, 2003)). The hypothesis is that we can eliminate some of the ambiguities (and consequent loss of subjectivity) observed during the lexicon translation by accounting for the context of the ambiguous words, which is possible in a corpus-based approach. Additionally, we also hope to improve the recall of the classifier, by addressing those cases not covered by the lexicon-based approach.

In the experiments reported in this section, we use a parallel corpus consisting of 107 documents from the SemCor corpus (Miller et al., 1993) and their manual translations into Romanian.[3] The corpus consists of roughly 11,000 sentences, with approximately 250,000 tokens on each side. It is a balanced corpus covering a number of topics in sports, politics, fashion, education, and others.

[3]The translation was carried out by a Romanian native speaker, student in a department of "Foreign Languages and Translations" in Romania.

Below, we begin with a manual annotation study to assess the quality of annotation and preservation of subjectivity in translation. We then describe the automatic construction of a target-language training set, and evaluate a classifier trained on that data.

Annotation Study.
We start by performing an agreement study meant to determine the extent to which subjectivity is preserved by the cross-lingual projections. In the study, three annotators – one native English speaker (*En*) and two native Romanian speakers (Ro_1 and Ro_2) – first trained on 3 randomly selected documents (331 sentences). They then independently annotated the subjectivity of the sentences of two randomly selected documents from the parallel corpus, accounting for 173 aligned sentence pairs. The annotators had access exclusively to the version of the sentences in their language, to avoid any bias that could be introduced by seeing the translation in the other language.

Note that the Romanian annotations (after all differences between the Romanian annotators were adjudicated) of all 331 + 173 sentences make up the gold standard corpus used in the experiments reported in Sections 3.2 and 4.1.

Before presenting the results of the annotation study, we give some examples. The following are English subjective sentences and their Romanian translations (the subjective elements are shown in bold).

> [en] **The desire to** give Broglio as many starts as possible.
> [ro] **Dorinţa de** a-i da lui Broglio cât mai multe starturi posibile.

> [en] **Suppose** he did lie beside Lenin, **would it be permanent ?**
> [ro] **Să presupunem** că ar fi aşezat alături de Lenin, **oare va fi pentru totdeauna?**

The following are examples of objective parallel sentences.

> [en]The Pirates have a 9-6 record this year and the Redbirds are 7-9.
> [ro] Piraţii au un palmares de 9 la 6 anul acesta si Păsările Roşii au 7 la 9.

> [en] One of the obstacles to the easy control of a 2-year old child is a lack of verbal communication.
> [ro] Unul dintre obstacolele în controlarea unui copil de 2 ani este lipsa comunicării verbale.

The annotators were trained using the MPQA annotation guidelines (Wiebe et al., 2005). The tagset consists of *S(ubjective)*, *O(bjective)* and *U(ncertain)*. For the *U* tags, a class was also given; *OU* means, for instance, that the annotator is uncertain but she is leaning toward *O*. Table 4 shows the pairwise agreement figures and the Kappa (κ) calculated for the three annotators. The table also shows the agreement when the borderline uncertain cases are removed.

	all sentences		Uncertain removed		
pair	agree	κ	agree	κ	(%) removed
Ro_1 & Ro_2	0.83	0.67	0.89	0.77	23
En & Ro_1	0.77	0.54	0.86	0.73	26
En & Ro_2	0.78	0.55	0.91	0.82	20

Table 4: Agreement on the data set of 173 sentences. Annotations performed by three annotators: one native English speaker (*En*) and two native Romanian speakers (Ro_1 and Ro_2)

When all the sentences are included, the agreement between the two Romanian annotators is measured at 0.83 (κ = 0.67). If we remove the borderline cases where at least one annotator's tag is *Uncertain*, the agreement rises to 0.89 with κ = 0.77. These figures are somewhat lower than the agreement observed during previous subjectivity annotation studies conducted on English (Wiebe et al., 2005) (the annotators were more extensively trained in those studies), but they nonetheless indicate consistent agreement.

Interestingly, when the agreement is conducted cross-lingually between an English and a Romanian annotator, the agreement figures, although somewhat lower, are comparable. In fact, once the *Uncertain* tags are removed, the monolingual and cross-lingual agreement and κ values become almost equal, which suggests that in most cases the sentence-level subjectivity is preserved.

The disagreements were reconciled first between the labels assigned by the two Romanian annotators, followed by a reconciliation between the resulting Romanian "gold-standard" labels and the labels assigned by the English annotator. In most cases, the disagreement across the two languages was found to be due to a difference of opinion about the sentence subjectivity, similar to the differences encountered in monolingual annotations. However, there are cases where the differences are due to the subjectivity being lost in the translation. Sometimes, this is due to several possible interpretations for the translated sentence. For instance, the following sentence:

> [en] They **honored** the battling Billikens last night.
> [ro] Ei i-au celebrat pe Billikens seara trecută.

is marked as *Subjective* in English (in context, the English annotator interpreted *honored* as referring to praises of the Billikens). However, the Romanian translation of *honored* is *celebrat* which, while correct as a translation, has the more frequent interpretation of *having a party.* The two Romanian annotators chose this interpretation, which correspondingly lead them to mark the sentence as *Objective*.

In other cases, in particular when the subjectivity is due to figures of speech such as irony, the translation sometimes misses the ironic aspects. For instance, the translation of *egghead* was not perceived as ironic by the Romanian annotators, and consequently the following sentence labeled *Subjective* in English is annotated as *Objective* in Romanian.

> [en] I have lived for many years in a Connecticut commuting town with a high percentage of [...] business executives of **egghead** tastes.
> [ro] Am trăit mulţi ani într-un oraş din apropiere de Connecticut ce avea o mare proporţie de [...] oameni de afaceri cu gusturi intelectuale.

4.1 Translating a Subjectivity-Annotated Corpus and Creating a Machine Learning Subjectivity Classifier

To further validate the corpus-based projection of subjectivity, we developed a subjectivity classifier trained on Romanian subjectivity-annotated corpora obtained via cross-lingual projections.

Ideally, one would generate an annotated Romanian corpus by translating English documents manually annotated for subjectivity such as the MPQA corpus. Unfortunately, the manual translation of this corpus would be prohibitively expensive, both timewise and financially. The other alternative – automatic machine translation – has not yet reached a level that would enable the generation of a high-quality translated corpus. We therefore decided to use a different approach where we automatically annotate the English side of an existing English-Romanian corpus, and subsequently project the annotations onto the Romanian side of the parallel cor-

	Precision	Recall	F-measure
high-precision	86.7	32.6	47.4
high-coverage	79.4	70.6	74.7

Table 5: Precision, recall, and F-measure for the two OpinionFinder classifiers, as measured on the MPQA corpus.

pus across the sentence-level alignments available in the corpus.

For the automatic subjectivity annotations, we generated two sets of the English-side annotations, one using the high-precision classifier and one using the high-coverage classifier available in the OpinionFinder tool. The high-precision classifier in OpinionFinder uses the clues of the subjectivity lexicon to harvest subjective and objective sentences from a large amount of unannotated text; this data is then used to automatically identify a set of extraction patterns, which are then used iteratively to identify a larger set of subjective and objective sentences.

In addition, in OpinionFinder, the high-precision classifier is used to produce an English labeled data set for training, which is used to generate its Naive Bayes high-coverage subjectivity classifier. Table 5 shows the performance of the two classifiers on the MPQA corpus as reported in (Wiebe and Riloff, 2005). Note that 55% of the sentences in the MPQA corpus are subjective – which represents the baseline for this data set.

The two OpinionFinder classifiers are used to label the training corpus. After removing the 504 test sentences, we are left with 10,628 sentences that are automatically annotated for subjectivity. Table 6 shows the number of subjective and objective sentences obtained with each classifier.

Classifier	Subjective	Objective	All
high-precision	1,629	2,334	3,963
high-coverage	5,050	5,578	10,628

Table 6: Subjective and objective training sentences automatically annotated with OpinionFinder.

Next, the OpinionFinder annotations are projected onto the Romanian training sentences, which are then used to develop a probabilistic classifier for the automatic labeling of subjectivity in Romanian sentences.

Similar to, e.g., (Pang et al., 2002), we use a Naive Bayes algorithm trained on word features co-occurring with the subjective and the objective classifications. We assume word independence, and we use a 0.3 cut-off for feature selection. While recent work has also considered more complex syntactic features, we are not able to generate such features for Romanian as they require tools currently not available for this language.

We create two classifiers, one trained on each data set. The quality of the classifiers is evaluated on the 504-sentence Romanian gold-standard corpus described above. Recall that the baseline on this data set is 54.16%, the percentage of sentences in the corpus that are subjective. Table 7 shows the results.

	Subjective	Objective	All
projection source: OF high-precision classifier			
Precision	65.02	69.62	64.48
Recall	82.41	47.61	64.48
F-measure	72.68	56.54	64.68
projection source: OF high-coverage classifier			
Precision	66.66	70.17	67.85
Recall	81.31	52.17	67.85
F-measure	72.68	56.54	67.85

Table 7: Evaluation of the machine learning classifier using training data obtained via projections from data automatically labeled by OpinionFinder (OF).

Our best classifier has an F-measure of 67.85, and is obtained by training on projections from the high-coverage OpinionFinder annotations. Although smaller than the 74.70 F-measure obtained by the English high-coverage classifier (see Table 5), the result appears remarkable given that no language-specific Romanian information was used.

The overall results obtained with the machine learning approach are considerably higher than those obtained from the rule-based classifier (except for the precision of the subjective sentences). This is most likely due to the lexicon translation process, which as mentioned in the agreement study in Section 3.1, leads to ambiguity and loss of subjectivity. Instead, the corpus-based translations seem to better account for the ambiguity of the words, and the subjectivity is generally preserved in the sentence translations.

5 Conclusions

In this paper, we described two approaches to generating resources for subjectivity annotations for a new

language, by leveraging on resources and tools available for English. The first approach builds a target language subjectivity lexicon by translating an existing English lexicon using a bilingual dictionary. The second generates a subjectivity-annotated corpus in a target language by projecting annotations from an automatically annotated English corpus.

These resources were validated in two ways. First, we carried out annotation studies measuring the extent to which subjectivity is preserved across languages in each of the two resources. These studies show that only a relatively small fraction of the entries in the lexicon preserve their subjectivity in the translation, mainly due to the ambiguity in both the source and the target languages. This is consistent with observations made in previous work that subjectivity is a property associated not with words, but with word *meanings* (Wiebe and Mihalcea, 2006). In contrast, the sentence-level subjectivity was found to be more reliably preserved across languages, with cross-lingual inter-annotator agreements comparable to the monolingual ones.

Second, we validated the two automatically generated subjectivity resources by using them to build a tool for subjectivity analysis in the target language. Specifically, we developed two classifiers: a rule-based classifier that relies on the subjectivity lexicon described in Section 3.1, and a machine learning classifier trained on the subjectivity-annotated corpus described in Section 4.1. While the highest precision for the subjective classification is obtained with the rule-based classifier, the overall best result of 67.85 F-measure is due to the machine learning approach. This result is consistent with the annotation studies, showing that the corpus projections preserve subjectivity more reliably than the lexicon translations.

Finally, neither one of the classifiers relies on language-specific information, but rather on knowledge obtained through projections from English. A similar method can therefore be used to derive tools for subjectivity analysis in other languages.

References

Alina Andreevskaia and Sabine Bergler. Mining wordnet for fuzzy sentiment: Sentiment tag extraction from WordNet glosses. In *Proceedings of EACL 2006.*

Cecilia Ovesdotter Alm, Dan Roth, and Richard Sproat. 2005. Emotions from text: Machine learning for text-based emotion prediction. In *Proceedings of HLT/EMNLP 2005.*

Krisztian Balog, Gilad Mishne, and Maarten de Rijke. 2006. Why are they excited? identifying and explaining spikes in blog mood levels. In *EACL-2006*.

Andrea Esuli and Fabrizio Sebastiani. 2006. Determining term subjectivity and term orientation for opinion mining. In *Proceedings the EACL 2006.*

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of ACM SIGKDD.*

Yi Hu, Jianyong Duan, Xiaoming Chen, Bingzhen Pei, and Ruzhan Lu. 2005. A new method for sentiment classification in text retrieval. In *Proceedings of IJCNLP.*

Radu Ion. 2007. *Methods for automatic semantic disambiguation. Applications to English and Romanian.* Ph.D. thesis, The Romanian Academy, RACAI.

Hiroshi Kanayama and Tetsuya Nasukawa. 2006. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *Proceedings of EMNLP 2006.*

Soo-Min Kim and Eduard Hovy. 2006. Identifying and analyzing judgment opinions. In *Proceedings of HLT/NAACL 2006.*

Levon Lloyd, Dimitrios Kechagias, and Steven Skiena. 2005. Lydia: A system for large-scale news analysis. In *Proceedings of SPIRE 2005.*

George Miller, Claudia Leacock, Tangee Randee, and Ross Bunker. 1993. A semantic concordance. In *Proceedings of the DARPA Workshop on Human Language Technology*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL 2004.*

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of EMNLP 2002.*

Ellen Riloff and Janyce Wiebe. 2003. Learning extraction patterns for subjective expressions. In *Proceedings of EMNLP 2003.*

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2006. Latent variable models for semantic orientations of phrases. In *Proceedings of EACL 2006.*

Peter Turney. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of ACL 2002.*

Universal Dictionary. 2007. Available at www.dicts.info/uddl.php.

Janyce Wiebe and Rada Mihalcea. 2006. Word sense and subjectivity. In *Proceedings of COLING-ACL 2006*.

Janyce Wiebe and Ellen Riloff. 2005. Creating subjective and objective sentence classifiers from unannotated texts. In *Proceedings of CICLing 2005 (invited paper).* Available at www.cs.pitt.edu/mpqarequest.

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. Available at www.cs.pitt.edu/mpqa.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT/EMNLP 2005.*

Hong Yu and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of EMNLP 2003.*

Sentiment Polarity Identification in Financial News: A Cohesion-based Approach

Ann Devitt
School of Computer Science & Statistics,
Trinity College Dublin, Ireland
Ann.Devitt@cs.tcd.ie

Khurshid Ahmad
School of Computer Science & Statistics,
Trinity College Dublin, Ireland
Khurshid.Ahmad@cs.tcd.ie

Abstract

Text is not unadulterated fact. A text can make you laugh or cry but can it also make you short sell your stocks in company A and buy up options in company B? Research in the domain of finance strongly suggests that it can. Studies have shown that both the informational and affective aspects of news text affect the markets in profound ways, impacting on volumes of trades, stock prices, volatility and even future firm earnings. This paper aims to explore a computable metric of positive or negative polarity in financial news text which is consistent with human judgments and can be used in a quantitative analysis of news sentiment impact on financial markets. Results from a preliminary evaluation are presented and discussed.

1 Introduction

Research in sentiment analysis has emerged to address the research questions: what is affect in text? what features of text serve to convey it? how can these features be detected and measured automatically. Sentence and phrase level sentiment analysis involves a systematic examination of texts, such as blogs, reviews and news reports, for positive, negative or neutral emotions (Wilson et al., 2005; Grefenstette et al., 2004). The term "sentiment analysis" is used rather differently in financial economics where it refers to the derivation of market confidence indicators from proxies such as stock prices and trading volumes. There is a tradition going back to the Nobel Sveriges–Riksbank Laureates Herbert Simon (1978 Prize) and Daniel Kahneman (2002 Prize), that shows that investors and traders in such markets can behave irrationally and that this bounded rationality is inspired by what the traders and investors hear from others about the conditions that may or may not prevail in the markets. Robert Engle (2003 Prize) has given a mathematical description of the asymmetric and affective impact of news on prices: positive news is typically related to large changes in prices but only for a short time; conversely the effect of negative news on prices and volumes of trading is longer lasting. The emergent domain of sociology of finance examines financial markets as social constructs and how communications, such as e-mails and news reports, may be loaded with sentiment which could distort market trading (MacKenzie, 2003).

It would appear that news affects the markets in profound ways, impacting on volumes of trade, stock returns, volatility of prices and even future firm earnings. In the domain of news impact analysis in finance, in recent years the focus has expanded from informational to affective content of text in an effort to explain the relationship between text and the markets. All text, be it news, blogs, accounting reports or poetry, has a non-factual dimension conveying opinion, invoking emotion, providing a nuanced perspective of the factual content of the text. With the increase of computational power and lexical and corpus resources it seems computationally feasible to detect some of the affective content of text automatically. The motivation for the work reported here is to identify a metric for sentiment po-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 984–991,
Prague, Czech Republic, June 2007.

larity which reliably replicates human evaluations and which is readily derivable from free text. This research is being carried out in the context of a study of the impact of news and its attendant biases on financial markets, formalizing earlier multi-lingual, corpus-based empirical work that analysed change in sentiment and volume of news in large financial news corpora (Ahmad et al., 2006). A systematic analysis of the impact of news bias or polarity on market variables requires a numeric value for sentiment intensity, as well as a binary tag for sentiment polarity, to identify trends in the sentiment indicator as well as turning points. In this approach, the contribution to an overall sentiment polarity and intensity metric of individual lexical items which are "affective" by definition is determined by their connectivity and position within a representation of the text as a whole based on the principles of lexical cohesion. The contribution of each element is therefore not purely additive but rather is mitigated by its relevance and position relative to other elements.

Section 2 sets out related work in the sentiment analysis domain both in computational linguistics and in finance where these techniques have been applied with some success. Section 3 outlines the cohesion-based algorithm for sentiment polarity detection, the resources used and the benefits of using the graph-based text representation approach. This approach was evaluated relative to a small corpus of gold standard sentiment judgments. The derivation of the gold standard and details of the evaluation are outlined in section 4. The results are presented and discussed in section 5 and section 6 concludes with a look at future challenges for this research.

2 Related Work

2.1 Cognitive Theories of Emotion

In order to understand how emotion can be realised in text, we must first have a notion of what emotion is and how people experience it. Current cognitive theories of what constitutes emotion are divided between two primary approaches: categorical and dimensional. The Darwinian categorical approach posits a finite set of basic emotions which are experienced universally across cultures, (e.g. anger, fear, sadness, surprise (Ekman and Friesen, 1971)). The second approach delineates emotions according to multiple dimensions rather than into discrete categories. The two primary dimensions in this account are a good–bad axis, the dimension of valence or evaluation, and a strong-weak axis, the dimension of activation or intensity (Osgood et al., 1957). The work reported here aims to conflate the evaluation and activation dimensions into one metric with the size of the value indicating strength of activation and its sign, polarity on the evaluation axis.

2.2 Sentiment Analysis

Sentiment analysis in computational linguistics has focused on examining what textual features (lexical, syntactic, punctuation, etc) contribute to affective content of text and how these features can be detected automatically to derive a sentiment metric for a word, sentence or whole text. Wiebe and colleagues have largely focused on identifying subjectivity in texts, i.e. identifying those texts which are affectively neutral and those which are not. This work has been grounded in a strong human evaluative component. The subjectivity identification research has moved from initial work using syntactic class, punctuation and sentence position features for subjectivity classifiers to later work using more lexical features like gradation of adjectives or word frequency (Wiebe et al., 1999; Wiebe et al., 2005). Others, such as Turney (2002), Pang and Vaithyanathan (2002), have examined the positive or negative polarity, rather than presence or absence, of affective content in text. Kim and Hovy (2004), among others, have combined the two tasks, identifying subjective text and detecting its sentiment polarity. The indicators of affective content have been drawn from lexical sources, corpora and the world wide web and combined in a variety of ways, including factor analysis and machine learning techniques, to determine when a text contains affective content and what is the polarity of that content.

2.3 Sentiment and News Impact Analysis

Niederhoffer (1971), academic and hedge fund manager, analysed 20 years of New York Times headlines classified into 19 semantic categories and on a good-bad rating scale to evaluate how the markets reacted to good and bed news: he found that markets do react to news with a tendency to overreact to bad news. Somewhat prophetically, he suggests

that news should be analysed by computers to introduce more objectivity in the analysis. Engle and Ng (1993) proposed the news impact curve as a model for how news impacts on volatility in the market with bad news introducing more volatility than good news. They used the market variable, stock returns, as a proxy for news, an unexpected drop in returns for bad news and an unexpected rise for good news. Indeed, much early work used such market variables or readily quantifiable aspects of news as a proxy for the news itself: e.g. news arrival, type, provenance and volumes (Cutler et al., 1989; Mitchell and Mulherin, 1994). More recent studies have proceeded in a spirit of computer-aided objectivity which entails determining linguistic features to be used to automatically categorise text into positive or negative news. Davis et al (2006) investigate the effects of optimistic or pessimistic language used in financial press releases on future firm performance. They conclude that a) readers form expectations regarding the habitual bias of writers and b) react more strongly to reports which violate these expectations, strongly suggesting that readers, and by extension the markets, form expectations about and react to not only content but also affective aspects of text. Tetlock (2007) also investigates how a pessimism factor, automatically generated from news text through term classification and principal components analysis, may forecast market activity, in particular stock returns. He finds that high negativity in news predicts lower returns up to 4 weeks around story release. The studies establish a relationship between affective bias in text and market activity that market players and regulators may have to address.

3 Approach

3.1 Cohesion-based Text Representation

The approach employed here builds on a cohesion-based text representation algorithm used in a news story comparison application described in (Devitt, 2004). The algorithm builds a graph representation of text from part-of-speech tagged text without disambiguation using WordNet (Fellbaum, 1998) as a real world knowledge source to reduce information loss in the transition from text to text-based structure. The representation is designed within the theoretical framework of lexical cohesion (Halliday and Hasan, 1976). Aspects of the cohesive structure of a text are captured in a graph representation which combines information derived from the text and WordNet semantic content. The graph structure is composed of nodes representing concepts in or derived from the text connected by relations between these concepts in WordNet, such as antonymy or hypernymy, or derived from the text, such as adjacency in the text. In addition, the approach provides the facility to manipulate or control how the WordNet semantic content information is interpreted through the use of topological features of the knowledge base. In order to evaluate the relative contribution of WordNet concepts to the information content of a text as a whole, a node specificity metric was derived based on an empirical analysis of the distribution of topological features of WordNet such as inheritance, hierarchy depth, clustering coefficients and node degree and how these features map onto human judgments of concept specificity or informativity. This metric addresses the issue of the uneven population of most knowledge bases so that the local idiosyncratic characteristics of WordNet can be mitigated by some of its global features.

3.2 Sentiment Polarity Overlay

By exploiting existing lexical resources for sentiment analysis, an explicit affective dimension can be overlaid on this basic text model. Our approach to polarity measurement, like others, relies on a lexicon of tagged positive and negative sentiment terms which are used to quantify positive/negative sentiment. In this first iteration of the work, SentiWN (Esuli and Sebastiani, 2006) was used as it provides a readily interpretable positive and negative polarity value for a set of "affective" terms which conflates Osgood's (1957) evaluative and activation dimensions. Furthermore, it is based on WordNet 2.0 and can therefore be integrated into the existing text representation algorithm, where some nodes in the cohesion graph carry a SentiWN sentiment value and others do not. The contribution of individual polarity nodes to the polarity metric of the text as a whole is then determined with respect to the textual information and WN semantic and topological features encoded in the underlying graph representation of the text. Three polarity metrics were implemented to evaluate the effectiveness of exploiting different

aspects of the cohesion-based graph structure.

Basic Cohesion Metric is based solely on frequency of sentiment-bearing nodes in *or derived from* the source text, i.e. the sum of polarity values for all nodes in the graph.

Relation Type Metric modifies the basic metric with respect to the types of WordNet relations in the text-derived graph. For each node in the graph, its sentiment value is the product of its polarity value and a relation weight for each relation this node enters into in the graph structure. Unlike most lexical chaining algorithms, not all WordNet relations are treated as equal. In this sentiment overlay, the relations which are deemed most relevant are those that potentially denote a relation of the affective dimension, like antonymy, and those which constitute key organising principles of the database, such as hypernymy. Potentially affect-effecting relations have the strongest weighting while more amorphous relations, such as "also see", have the lowest.

Node Specificity Metric modifies the basic metric with respect to a measure of node specificity calculated on the basis of topographical features of WordNet. The intuition behind this measure is that highly specific nodes or concepts may carry more informational and, by extension, affective content than less specific ones. We have noted the difficulty of using a knowledge base whose internal structure is not homogeneous and whose idiosyncrasies are not quantified. The specificity measure aims to factor out population sparseness or density in WordNet by evaluating the contribution of each node relative to its depth in the hierarchy, its connectivity (branchingFactor) and its siblings:

$$Spc = \frac{(depth + \ln(siblings) - ln(branchingFactor))}{NormalizingFactor} \quad (1)$$

The three metrics are further specialised according to the following two boolean flags:

InText: the metric is calculated based on 1) only those nodes representing terms in the source text, or 2) all nodes in the graph representation derived from the text. In this way, the metrics can be calculated using information derived from the graph representation, such as node specificity, without potentially noisy contributions from nodes not in the source text but related to them, via relations such as hypernymy.

Modifiers: the metric is calculated using all open class parts of speech or modifiers alone. On a cursory inspection of SentiWN, it seems that modifiers have more reliable values than nouns or verbs. This option was included to test for possible adverse effects of the lexicon.

In total for each metric there are four outcomes combining **inText** true/false and **modifiers** true/false.

4 Evaluation

The goal of this research is to examine the relationship between financial markets and financial news, in particular the polarity of financial news. The domain of finance provides data and methods for solid quantitative analysis of the impact of sentiment polarity in news. However, in order to engage with this long tradition of analysis of the instruments and related variables of the financial markets, the quantitative measure of polarity must be not only easy to compute, it must be consistent with human judgments of polarity in this domain. This evaluation is a first step on the path to establishing reliability for a sentiment measure of news. Unfortunately, the focus on news, as opposed to other text types, has its difficulties. Much of the work in sentiment analysis in the computational linguistics domain has focused either on short segments, such as sentences (Wilson et al., 2005), or on longer documents with an explicit polarity orientation like movie or product reviews (Turney, 2002). Not all news items may express overt sentiment. Therefore, in order to test our hypothesis, we selected a news topic which was considered a priori to have emotive content.

4.1 Corpus

Markets react strongest to information about firms to which they have an emotional attachment (MacGregor et al., 2000). Furthermore, takeovers and mergers are usually seen as highly emotive contexts. To combine these two emotion-enhancing factors, a corpus of news texts was compiled on the topic of the aggressive takeover bid of a low-cost airline (Ryanair) for the Irish flag-carrier airline (Aer Lingus). Both airlines have a strong (positive and negative) emotional attachment for many in Ireland. Furthermore, both airlines are highly visible within the country and have vocal supporters and detractors in the public arena. The corpus is drawn from the

national media and international news wire sources and spans 4 months in 2006 from the flotation of the flag carrier on the stock exchange in September 2006, through the "surprise" take-over bid announcement by Ryanair, to the withdrawal of the bid by Ryanair in December 2006.[1]

4.2 Gold Standard

A set of 30 texts selected from the corpus was annotated by 3 people on a 7-point scale from *very positive* to *very negative*. Given that a takeover bid has two players, the respondents were asked also to rate the semantic orientation of the texts with respect to the two players, Ryanair and Aer Lingus. Respondents were all native English speakers, 2 female and 1 male. To ensure emotional engagement in the task, they were first asked to rate their personal attitude to the two airlines. The ratings in all three cases were on the extreme ends of the 7 point scale, with very positive attitudes towards the flag carrier and very negative attitudes towards the low-cost airline. Respondent attitudes may impact on their text evaluations but, given the high agreement of attitudes in this study, this impact should at least be consistent across the individuals in the study. A larger study should control explicitly for this variable.

As the respondents gave ratings on a ranked scale, inter-respondent reliability was determined using Krippendorf's alpha, a modification of the Kappa coefficient for ordinal data (Krippendorff, 1980). On the general ranking scale, there was little agreement ($kappa = 0.1685$), corroborating feedback from respondents on the difficulty of providing a general rating for text polarity distinct from a rating with respect to one of the two companies. However, there was an acceptable degree of agreement (Grove et al., 1981) on the Ryanair and Aer Lingus polarity ratings, $kappa = 0.5795$ and $kappa = 0.5589$ respectively. Results report correlations with these ratings which are consistent and, from the financial market perspective, potentially more interesting.[2]

[1] A correlation analysis of human sentiment ratings with Ryanair and Aer Lingus stock prices for the last quarter of 2006 was conducted. The findings suggest that stock prices were correlated with ratings with respect to Aer Lingus, suggesting that, during this takeover period, investors may have been influenced by sentiment expressed in news towards Aer Lingus. However, the timeseries is too short to ensure statistical significance.

[2] Results in this paper are reported with respect to the Ryanair ratings as they had the highest inter-rater agreement.

4.3 Performance Metrics

The performance of the polarity algorithm was evaluated relative to a corpus of human-annotated news texts, focusing on two separate dimensions of polarity:

1. Polarity direction: the task of assigning a binary positive/negative value to a text
2. Polarity intensity: the task of assigning a value to indicate the strength of the negative/positive polarity in a text.

Performance on the former is reported using standard recall and precision metrics. The latter is reported as a correlation with average human ratings.

4.4 Baseline

For the metrics in section 3, the baseline for comparison sums the SentiWN polarity rating for only those lexical items present in the text, not exploiting any aspect of the graph representation of the text. This baseline corresponds to the Basic Cohesion Metric, with $inText = true$ (only lexical items in the text) and $modifiers = false$ (all parts of speech).

5 Results and Discussion

5.1 Binary Polarity Assignment

The baseline results for positive ratings, negative ratings and overall accuracy for the task of assigning a polarity tag are reported in table 1. The results show

Type	Precision	Recall	FScore
Positive	0.381	0.7273	0.5
Negative	0.667	0.3158	0.4286
Overall	0.4667	0.4667	0.4667

Table 1: Baseline results

that the baseline tends towards the positive end of the rating spectrum, with high recall for positive ratings but low precision. Conversely, negative ratings have high precision but low recall. Figures 1 to 3 illustrate the performance for positive, negative and overall ratings of all metric–inText–Modifier combinations, enumerated in table 2, relative to this baseline, the horizontal. Those metrics which surpass this line are deemed to outperform the baseline.

1	Cohesion	**5**	Relation	**9**	NodeSpec
2	CohesionTxt	**6**	RelationTxt	**10**	NodeSpecTxt
3	CohesionMod	**7**	RelationMod	**11**	NodeSpecMod
4	CohesionTxtMod	**8**	RelationTxtMod	**12**	NodeSpecTxtMod

Table 2: Metric types in Figures 1-3

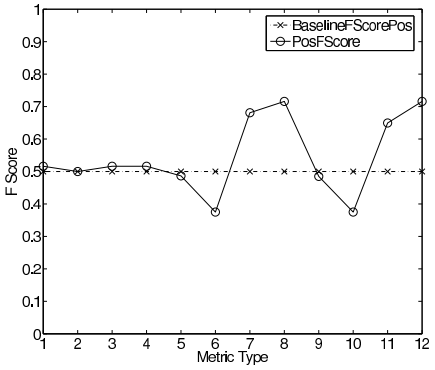


Figure 1: F Score for Positive Ratings

All metrics have a bias towards positive ratings with attendant high positive recall values and improved f-score for positive polarity assignments. The Basic Cohesion Metric marginally outperforms the baseline overall indicating that exploiting the graph structure gives some added benefit. For the Relations and Specificity metrics, system performance greatly improves on the baseline for the $modifiers = true$ options, whereas, when all parts of speech are included ($modifier = false$), performance drops significantly. This sensitivity to inclusion of all word classes could suggest that modifiers are better indicators of text polarity than other word classes or that the metrics used are not appropriate to non-modifier parts of speech. The former hypothesis is not supported by the literature while the latter is not supported by prior successful application of these metrics in a text comparison task. In order to investigate the source of this sensitivity, we intend to examine the distribution of relation types and node specificity values for sentiment-bearing terms to determine how best to tailor these metrics to the sentiment identification task.

A further hypothesis is that the basic polarity values for non-modifiers are less reliable than for adjectives and adverbs. On a cursory inspection of polarity values of nouns and adjectives in SentiWN, it would appear that adjectives are somewhat more reliably labelled than nouns. For example, crime and

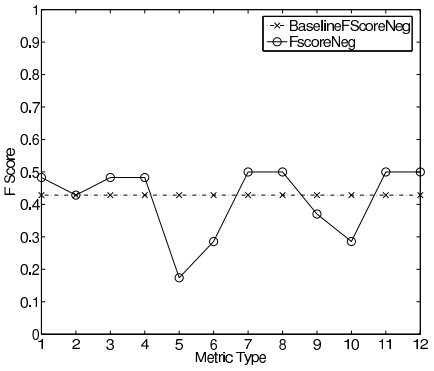


Figure 2: F Score for Negative Ratings

some of its hyponyms are labelled as neutral (e.g. forgery) or even positive (e.g. assault) whereas criminal is labelled as negative. This illustrates a key weakness in a lexical approach such as this: over-reliance on lexical resources. No lexical resource is infallible. It is therefore vital to spread the associated risk by using more than one knowledge source, e.g. multiple sentiment lexica or using corpus data.

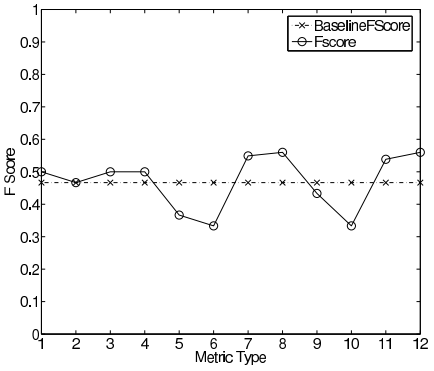


Figure 3: F Score for All Ratings

5.2 Polarity Intensity Values

The results on the polarity intensity task parallel the results on polarity tag assignment. Table 3 sets out the correlation coefficients for the metrics with respect to the average human rating. Again, the best performers are the relation type and node specificity metrics using only modifiers, significant to the 0.05 level. Yet the correlation coefficients overall are not very high. This would suggest that perhaps the relationship between the human ranking scale and the automatic one is not strictly linear. Although the human ratings map approximately onto the automati-

cally derived scale, there does not seem to be a clear one to one mapping. The section that follows discuss this and some of the other issues which this evaluation process has brought to light.

Metric	inText	Modifier	Correlation
Basic Cohesion	No	No	0.47**
	Yes	No	0.42*
	No	Yes	0.47**
	Yes	Yes	0.47**
Relation Type	No	No	-0.1**
	Yes	No	-0.13*
	No	Yes	0.5**
	Yes	Yes	0.38*
Node Specificity	No	No	0.00
	Yes	No	-0.03
	No	Yes	0.48**
	Yes	Yes	0.38*

Table 3: Correlation Coefficients for human ratings.
**. Significant at the 0.01 level. *. Significant at the 0.05 level.

5.3 Issues

The Rating Scale and Thresholding

Overall the algorithm tends towards the positive end of the spectrum in direct contrast to human raters with 55-70% of all ratings being negative. Furthermore, the correlation of human to algorithm ratings is significant but not strongly directional. It would appear that there are more positive lexical items in text, hence the algorithm's positive bias. Yet much of this positivity is not having a strong impact on readers, hence the negative bias observed in these evaluators. This raises questions about the scale of human polarity judgments: are people more sensitive to negativity in text? is there a positive baseline in text that people find unremarkable and ignore? To investigate this issue, we will conduct a comparative corpus analysis of the distribution of positive and negative lexical items in text and their perceived strengths in text. The results of this analysis should help to locate sentiment turning points or thresholds and establish an elastic sentiment scale which allows for baseline but disregarded positivity in text.

The Impact of the Lexicon

The algorithm described here is lexicon-based, fully reliant on available lexical resources. However, we have noted that an over-reliance on lexica has its disadvantages, as any hand-coded or corpus-derived lexicon will have some degree of error or inconsistency. In order to address this issue, it is necessary to spread the risk associated with a single lexical resource by drawing on multiple sources, as in (Kim and Hovy, 2005). The SentiWN lexicon used in this implementation is derived from a seed word set supplemented WordNet relations and as such it has not been psychologically validated. For this reason, it has good coverage but some inconsistency. Whissel's Dictionary of Affect (1989) on the other hand is based entirely on human ratings of terms. It's coverage may be narrower but accuracy might be more reliable. This dictionary also has the advantage of separating out Osgood's (1957) evaluative and activation dimensions as well as an "imaging" rating for each term to allow a multi-dimensional analysis of affective content. The WN Affect lexicon (Valitutti et al., 2004) again provides somewhat different rating types where terms are classified in terms of denoting or evoking different physical or mental affective reactions. Together, these resources could offer not only more accurate base polarity values but also more nuanced metrics that may better correspond to human notions of affect in text.

The Gold Standard

Sentiment rating evaluation is not a straight-forward task. Wiebe et al (2005) note many of the difficulties associated human sentiment ratings of text. As noted above, it can be even more difficult when evaluating news where the text is intended to appear impartial. The attitude of the evaluator can be all important: their attitude to the individuals or organisations in the text, their professional viewpoint as a market player or an ordinary punter, their attitude to uncertainty and risk which can be a key factor in the world of finance. In order to address these issues for the domain of news impact in financial markets, the expertise of market professionals must be elicited to determine what they look for in text and what viewpoint they adopt when reading financial news. In econometric analysis, stock price or trading volume data constitute an alternative gold standard, representing a proxy for human reaction to news. For economic significance, the data must span a time period of several years and compilation of a text and stock

price corpus for a large scale analysis is underway.

6 Conclusions and Future Work

This paper presents a lexical cohesion based metric of sentiment intensity and polarity in text and an evaluation of this metric relative to human judgments of polarity in financial news. We are conducting further research on how best to capture a psychologically plausible measure of affective content of text by exploiting available resources and a broader evaluation of the measure relative to human judgments and existing metrics. This research is expected to contribute to sentiment analysis in finance. Given a reliable metric of sentiment in text, what is the impact of changes in this value on market variables? This involves a sociolinguistic dimension to determine what publications or texts best characterise or are most read and have the greatest influence in this domain and the economic dimension of correlation with economic indicators.

References

Khurshid Ahmad, David Cheng, and Yousif Almas. 2006. Multi–lingual sentiment analysis in financial news streams. In *Proc. of the 1st Intl. Conf. on Grid in Finance*, Italy.

David M. Cutler, James M. Poterba, and Lawrence H. Summers. 1989. What moves stock prices. *Journal of Portfolio Management*, 79:223–260.

Angela K. Davis, Jeremy M. Piger, and Lisa M. Sedor. 2006. Beyond the numbers: An analysis of optimistic and pessimistic language in earnings press releases. Technical report, Federal Reserve Bank of St Louis.

Ann Devitt. 2004. *Methods for Meaningful Text Representation and Comparison*. Ph.D. thesis, Trinity College Dublin.

Paul Ekman and W. V. Friesen. 1971. Constants across cultures in the face and emotion. *Journal of Personality and Social Psychology*, 17:124–129.

Robert F. Engle and Victor K. Ng. 1993. Measuring and testing the impact of news on volatility. *Journal of Finance*, 48(5):1749–1778.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of LREC 2006*.

Christiane Fellbaum. 1998. *WordNet,an electronic lexical database*. MIT Press.

Gregory Grefenstette, Yan Qu, James G. Shanahan, and David A. Evans. 2004. Coupling niche browsers and affect analysis for an opinion mining application. In *Proceedings of RIAO-04*, pages 186–194.

William N. Grove, Nancy C. Andreasen, Patricia McDonald-Scott, Martin B. Keller, and Robert W. Shapiro. 1981. Reliability studies of psychiatric diagnosis. theory and practice. *Archives of General Psychiatry*, 38:408–413.

Michael A. K. Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*. Longman.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of COLING 2004*.

Soo-Min Kim and Eduard Hovy. 2005. Automatic detection of opinion bearing words and sentences. In *Proc. of IJCNLP-05*, Jeju Island, Korea.

Klaus Krippendorff. 1980. *Content Analysis: an Introduction to its Methodology*. Sage Publications, Beverly Hills, CA.

Donald G. MacGregor, Paul Slovic, David Dreman, and Michael Berry. 2000. Imagery, affect, and financial judgment. *The Journal of Psychology and Financial Markets*, 1(2):104–110.

Donald MacKenzie. 2003. Long-term capital management and the sociology of arbitrage. *Economy and Society*, 32:349–380.

Mark L. Mitchell and J. Harold Mulherin. 1994. The impact of public information on the stock market. *Journal of Finance*, 49(3):923–950.

Victor Niederhoffer. 1971. The analysis of world events and stock prices. *Journal of Business*, 44(2):193–219.

Charles E. Osgood, George J. Suci, and Percy H. Tannenbaum. 1957. *The Measurement of meaning*. University of Illinois Press, Chicago, Ill.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proc. of EMNLP-02*, pages 79–86.

Paul C. Tetlock. 2007. Giving content to investor sentiment: The role of media in the stock market. *Journal of Finance*. forthcoming.

Peter D. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of ACL'02*, pages 417–424.

Alessandro Valitutti, Carlo Strapparava, and Oliviero Stock. 2004. Developing affective lexical resources. *PsychNology Journal*, 2(1):61–83.

Cynthia Whissell. 1989. The dictionary of affect in language. In R. Plutchik and H. Kellerman, editors, *Emotion: theory research and experience*, volume 4. Acad. Press, London.

Janyce M. Wiebe, Rebecca F. Bruce, and Thomas P. O'Hara. 1999. Development and use of a gold-standard data set for subjectivity classifications. In *Proceedings of ACL-99*.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39:165–210.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proc. of HLT/EMNLP-2005*, pages 347–354.

Weakly Supervised Learning for Hedge Classification in Scientific Literature

Ben Medlock
Computer Laboratory
University of Cambridge
Cambridge, CB3 OFD
benmedlock@cantab.net

Ted Briscoe
Computer Laboratory
University of Cambridge
Cambridge, CB3 OFD
ejb@cl.cam.ac.uk

Abstract

We investigate automatic classification of speculative language ('hedging'), in biomedical text using weakly supervised machine learning. Our contributions include a precise description of the task with annotation guidelines, analysis and discussion, a probabilistic weakly supervised learning model, and experimental evaluation of the methods presented. We show that hedge classification is feasible using weakly supervised ML, and point toward avenues for future research.

1 Introduction

The automatic processing of scientific papers using NLP and machine learning (ML) techniques is an increasingly important aspect of technical informatics. In the quest for a deeper machine-driven 'understanding' of the mass of scientific literature, a frequently occuring linguistic phenomenon that must be accounted for is the use of *hedging* to denote propositions of a speculative nature. Consider the following:

1. *Our results prove that XfK89 inhibits Felin-9.*
2. *Our results suggest that XfK89 might inhibit Felin-9.*

The second example contains a hedge, signaled by the use of *suggest* and *might*, which renders the proposition *inhibit(XfK89→Felin-9)* speculative. Such analysis would be useful in various applications; for instance, consider a system designed to identify and extract interactions between genetic entities in the biomedical domain. Case 1 above provides clear textual evidence of such an interaction and justifies extraction of *inhibit(XfK89→Felin-9)*, whereas case 2 provides only weak evidence for such an interaction.

Hedging occurs across the entire spectrum of scientific literature, though it is particularly common in the experimental natural sciences. In this study we consider the problem of learning to automatically classify sentences containing instances of hedging, given only a very limited amount of annotator-labelled 'seed' data. This falls within the *weakly supervised* ML framework, for which a range of techniques have been previously explored. The contributions of our work are as follows:

1. We provide a clear description of the problem of hedge classification and offer an improved and expanded set of annotation guidelines, which as we demonstrate experimentally are sufficient to induce a high level of agreement between independent annotators.
2. We discuss the specificities of hedge classification as a weakly supervised ML task.
3. We derive a probabilistic weakly supervised learning model and use it to motivate our approach.
4. We analyze our learning model experimentally and report promising results for the task on a new publicly-available dataset.[1]

2 Related Work

2.1 Hedge Classification

While there is a certain amount of literature within the linguistics community on the use of hedging in

[1] available from www.cl.cam.ac.uk/∼bwm23/

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 992–999,
Prague, Czech Republic, June 2007.

scientific text, eg. (Hyland, 1994), there is little of direct relevance to the task of classifying speculative language from an NLP/ML perspective.

The most clearly relevant study is Light et al. (2004) where the focus is on introducing the problem, exploring annotation issues and outlining potential applications rather than on the specificities of the ML approach, though they do present some results using a manually crafted substring matching classifier and a supervised SVM on a collection of *Medline* abstracts. We will draw on this work throughout our presentation of the task.

Hedging is sometimes classed under the umbrella concept of *subjectivity*, which covers a variety of linguistic phenomena used to express differing forms of authorial opinion (Wiebe et al., 2004). Riloff et al. (2003) explore bootstrapping techniques to identify subjective nouns and subsequently classify subjective vs. objective sentences in newswire text. Their work bears some relation to ours; however, our domains of interest differ (newswire vs. scientific text) and they do not address the problem of hedge classification directly.

2.2 Weakly Supervised Learning

Recent years have witnessed a significant growth of research into weakly supervised ML techniques for NLP applications. Different approaches are often characterised as either *multi-* or *single-view*, where the former generate multiple redundant (or semi-redundant) 'views' of a data sample and perform mutual bootstrapping. This idea was formalised by Blum and Mitchell (1998) in their presentation of *co-training*. Co-training has also been used for named entity recognition (NER) (Collins and Singer, 1999), coreference resolution (Ng and Cardie, 2003), text categorization (Nigam and Ghani, 2000) and improving gene name data (Wellner, 2005).

Conversely, single-view learning models operate without an explicit partition of the feature space. Perhaps the most well known of such approaches is *expectation maximization* (EM), used by Nigam et al. (2000) for text categorization and by Ng and Cardie (2003) in combination with a meta-level feature selection procedure. *Self-training* is an alternative single-view algorithm in which a labelled pool is incrementally enlarged with unlabelled samples for which the learner is most confident. Early work by Yarowsky (1995) falls within this framework. Banko and Brill (2001) use 'bagging' and agreement to measure confidence on unlabelled samples, and more recently McClosky et al. (2006) use self-training for improving parse reranking.

Other relevant recent work includes (Zhang, 2004), in which random feature projection and a committee of SVM classifiers is used in a hybrid co/self-training strategy for weakly supervised relation classification and (Chen et al., 2006) where a graph based algorithm called *label propagation* is employed to perform weakly supervised relation extraction.

3 The Hedge Classification Task

Given a collection of sentences, $\mathcal{S}$, the task is to label each sentence as either speculative or non-speculative (*spec* or *nspec* henceforth). Specifically, $\mathcal{S}$ is to be partitioned into two disjoint sets, one representing sentences that contain some form of hedging, and the other representing those that do not.

To further elucidate the nature of the task and improve annotation consistency, we have developed a new set of guidelines, building on the work of Light et al. (2004). As noted by Light et al., speculative assertions are to be identified on the basis of judgements about the author's intended meaning, rather than on the presence of certain designated hedge terms.

We begin with the hedge definition given by Light et al. (item 1) and introduce a set of further guidelines to help elucidate various 'grey areas' and tighten the task specification. These were developed after initial annotation by the authors, and through discussion with colleagues. Further examples are given in online Appendix A[2].

The following are considered hedge instances:

1. An assertion relating to a result that does not necessarily follow from work presented, but could be extrapolated from it (Light et al.).
2. Relay of hedge made in previous work.

   Dl and Ser have been proposed to act redundantly in the sensory bristle lineage.
3. Statement of knowledge paucity.

[2] available from www.cl.cam.ac.uk/∼bwm23/

How endocytosis of Dl leads to the activation of N remains to be elucidated.

4. Speculative question.

   A second important question is whether the roX genes have the same, overlapping or complementing functions.

5. Statement of speculative hypothesis.

   To test whether the reported sea urchin sequences represent a true RAG1-like match, we repeated the BLASTP search against all GenBank proteins.

6. Anaphoric hedge reference.

   This hypothesis is supported by our finding that both pupariation rate and survival are affected by EL9.

The following are not considered hedge instances:

1. Indication of experimentally observed non-universal behaviour.

   proteins with single BIR domains can also have functions in cell cycle regulation and cytokinesis.

2. Confident assertion based on external work.

   Two distinct E3 ubiquitin ligases have been shown to regulate Dl signaling in Drosophila melanogaster.

3. Statement of existence of proposed alternatives.

   Different models have been proposed to explain how endocytosis of the ligand, which removes the ligand from the cell surface, results in N receptor activation.

4. Experimentally-supported confirmation of previous speculation.

   Here we show that the hemocytes are the main regulator of adenosine in the Drosophila larva, as was speculated previously for mammals.

5. Negation of previous hedge.

   Although the adgf-a mutation leads to larval or pupal death, we have shown that this is not due to the adenosine or deoxyadenosine simply blocking cellular proliferation or survival, as the experiments in vitro would suggest.

4 Data

We used an archive of 5579 full-text papers from the functional genomics literature relating to Drosophila melanogaster (the fruit fly). The papers were converted to XML and linguistically processed using the RASP toolkit[3]. We annotated six of the papers to form a test set with a total of 380 *spec* sentences and 1157 *nspec* sentences, and randomly selected 300,000 sentences from the remaining papers as training data for the weakly supervised learner. To ensure selection of complete sentences rather than headings, captions etc., unlabelled samples were chosen under the constraints that they must be at least 10 words in length and contain a main verb.

[3] www.informatics.susx.ac.uk/research/nlp/rasp

	F_1^{rel}	κ
Original	0.8293	0.9336
Corrected	0.9652	0.9848

Table 1: Agreement Scores

5 Annotation and Agreement

Two separate annotators were commissioned to label the sentences in the test set, firstly one of the authors and secondly a domain expert with no prior input into the guideline development process. The two annotators labelled the data independently using the guidelines outlined in section 3. Relative F_1 (F_1^{rel}) and *Cohen's Kappa* (κ) were then used to quantify the level of agreement. For brevity we refer the reader to (Artstein and Poesio, 2005) and (Hripcsak and Rothschild, 2004) for formulation and discussion of κ and F_1^{rel} respectively.

The two metrics are based on different assumptions about the nature of the annotation task. F_1^{rel} is founded on the premise that the task is to recognise and label *spec* sentences from within a background population, and does not explicitly model agreement on *nspec* instances. It ranges from 0 (no agreement) to 1 (no disagreement). Conversely, κ gives explicit credit for agreement on both *spec* and *nspec* instances. The observed agreement is then corrected for 'chance agreement', yielding a metric that ranges between -1 and 1. Given our definition of hedge classification and assessing the manner in which the annotation was carried out, we suggest that the founding assumption of F_1^{rel} fits the nature of the task better than that of κ.

Following initial agreement calculation, the instances of disagreement were examined. It turned out that the large majority of cases of disagreement were due to negligence on behalf of one or other of the annotators (i.e. cases of clear hedging that were missed), and that the cases of genuine disagreement were actually quite rare. New labelings were then created with the negligent disagreements corrected, resulting in significantly higher agreement scores. Values for the original and negligence-corrected la-

belings are reported in Table 1.

Annotator conferral violates the fundamental assumption of annotator independence, and so the latter agreement scores do not represent the true level of agreement; however, it is reasonable to conclude that the actual agreement is approximately lower bounded by the initial values and upper bounded by the latter values. In fact even the lower bound is well within the range usually accepted as representing 'good' agreement, and thus we are confident in accepting human labeling as a gold-standard for the hedge classification task. For our experiments, we use the labeling of the genetics expert, corrected for negligent instances.

6 Discussion

In this study we use single terms as features, based on the intuition that many hedge cues are single terms (*suggest*, *likely* etc.) and due to the success of 'bag of words' representations in many classification tasks to date. Investigating more complex sample representation strategies is an avenue for future research.

There are a number of factors that make our formulation of hedge classification both interesting and challenging from a weakly supervised learning perspective. Firstly, due to the relative sparsity of hedge cues, most samples contain large numbers of irrelevant features. This is in contrast to much previous work on weakly supervised learning, where for instance in the case of text categorization (Blum and Mitchell, 1998; Nigam et al., 2000) almost all content terms are to some degree relevant, and irrelevant terms can often be filtered out (e.g. stopword removal). In the same vein, for the case of entity/relation extraction and classification (Collins and Singer, 1999; Zhang, 2004; Chen et al., 2006) the context of the entity or entities in consideration provides a highly relevant feature space.

Another interesting factor in our formulation of hedge classification is that the *nspec* class is defined on the basis of the *absence* of hedge cues, rendering it hard to model directly. This characteristic is also problematic in terms of selecting a reliable set of *nspec* seed sentences, as by definition at the beginning of the learning cycle the learner has little knowledge about what a hedge looks like. This problem is addressed in section 10.3.

In this study we develop a learning model based around the concept of iteratively predicting labels for unlabelled training samples, the basic paradigm for both co-training and self-training. However we generalise by framing the task in terms of the acquisition of labelled training data, from which a supervised classifier can subsequently be learned.

7 A Probabilistic Model for Training Data Acquisition

In this section, we derive a simple probabilistic model for acquiring training data for a given learning task, and use it to motivate our approach to weakly supervised hedge classification.

Given:

- sample space $\mathcal{X}$
- set of target concept classes $\mathcal{Y} = \{y_1 \dots y_N\}$
- target function $Y : \mathcal{X} \rightarrow \mathcal{Y}$
- set of seed samples for each class $\mathcal{S}_1 \dots \mathcal{S}_N$ where $\mathcal{S}_i \subset \mathcal{X}$ and $\forall \mathbf{x} \in \mathcal{S}_i[Y(\mathbf{x}) = y_i]$
- set of unlabelled samples $\mathcal{U} = \{\mathbf{x}_1 \dots \mathbf{x}_K\}$

Aim: *Infer a set of training samples $\mathcal{T}_i$ for each concept class y_i such that* $\forall \mathbf{x} \in \mathcal{T}_i[Y(\mathbf{x}) = y_i]$

Now, it follows that $\forall \mathbf{x} \in \mathcal{T}_i[Y(\mathbf{x}) = y_i]$ is satisfied in the case that $\forall \mathbf{x} \in \mathcal{T}_i[P(y_i|x) = 1]$, which leads to a model in which $\mathcal{T}_i$ is initialised to $\mathcal{S}_i$ and then iteratively augmented with the unlabelled sample(s) for which the posterior probability of class membership is maximal. Formally:

At each iteration:

$$\mathcal{T}_i \leftarrow \mathbf{x}_j (\in \mathcal{U}) \quad \text{where } j = \arg\max_j [P(y_i|\mathbf{x}_j)] \tag{1}$$

Expansion with Bayes' Rule yields:

$$\arg\max_j [P(y_i|\mathbf{x}_j)] = \arg\max_j \left[\frac{P(\mathbf{x}_j|y_i) \cdot P(y_i)}{P(\mathbf{x}_j)} \right] \tag{2}$$

An interesting observation is the importance of the sample prior $P(\mathbf{x}_j)$ in the denominator, often ignored for classification purposes because of its invariance to class. We can expand further by

marginalising over the classes in the denominator in expression 2, yielding:

$$\arg\max_j \left[\frac{P(\mathbf{x}_j|y_i) \cdot P(y_i)}{\sum_{n=1}^{N} P(y_n)P(\mathbf{x}_j|y_n)} \right] \quad (3)$$

so we are left with the class priors and class-conditional likelihoods, which can usually be estimated directly from the data, at least under limited dependence assumptions. The class priors can be estimated based on the relative distribution sizes derived from the current training sets:

$$P(y_i) = \frac{|\mathcal{T}_i|}{\sum_k |\mathcal{T}_k|} \quad (4)$$

where $|\mathcal{S}|$ is the number of samples in training set $\mathcal{S}$.

If we assume feature independence, which as we will see for our task is not as gross an approximation as it may at first seem, we can simplify the class-conditional likelihood in the well known manner:

$$P(\mathbf{x}_j|y_i) = \prod_k P(x_{jk}|y_i) \quad (5)$$

and then estimate the likelihood for each feature:

$$P(x_k|y_i) = \frac{\alpha P(y_i) + f(x_k, \mathcal{T}_i)}{\alpha P(y_i) + |\mathcal{T}_i|} \quad (6)$$

where $f(x, \mathcal{S})$ is the number of samples in training set $\mathcal{S}$ in which feature x is present, and α is a universal smoothing constant, scaled by the class prior. This scaling is motivated by the principle that without knowledge of the true distribution of a particular feature it makes sense to include knowledge of the class distribution in the smoothing mechanism. Smoothing is particularly important in the early stages of the learning process when the amount of training data is severely limited resulting in unreliable frequency estimates.

8 Hedge Classification

We will now consider how to apply this learning model to the hedge classification task. As discussed earlier, the speculative/non-speculative distinction hinges on the presence or absence of a few hedge cues within the sentence. Working on this premise, all features are ranked according to their probability of 'hedge cue-ness':

$$P(spec|x_k) = \frac{P(x_k|spec) \cdot P(spec)}{\sum_{n=1}^{N} P(y_n)P(x_k|y_n)} \quad (7)$$

which can be computed directly using (4) and (6). The m most probable features are then selected from each sentence to compute (5) and the rest are ignored. This has the dual benefit of removing irrelevant features and also reducing dependence between features, as the selected features will often be non-local and thus not too tightly correlated.

Note that this idea differs from traditional feature selection in two important ways:

1. Only features indicative of the *spec* class are retained, or to put it another way, *nspec* class membership is inferred from the absence of strong *spec* features.
2. Feature selection in this context is *not* a preprocessing step; i.e. there is no re-estimation after selection. This has the potentially detrimental side effect of skewing the posterior estimates in favour of the *spec* class, but is admissible for the purposes of ranking and classification by posterior thresholding (see next section).

9 Classification

The weakly supervised learner returns a labelled data set for each class, from which a classifier can be trained. We can easily derive a classifier using the estimates from our learning model by:

$$\mathbf{x}_j \rightarrow spec \quad \text{if} \quad P(spec|\mathbf{x}_j) > \sigma \quad (8)$$

where σ is an arbitrary threshold used to control the precision/recall balance. For comparison purposes, we also use Joachims' SVM^{light} (Joachims, 1999).

10 Experimental Evaluation

10.1 Method

To examine the practical efficacy of the learning and classification models we have presented, we use the following experimental method:

1. Generate seed training data: $\mathcal{S}_{spec}$ and $\mathcal{S}_{nspec}$
2. Initialise: $\mathcal{T}_{spec} \leftarrow \mathcal{S}_{spec}$ and $\mathcal{T}_{nspec} \leftarrow \mathcal{S}_{nspec}$
3. Iterate:
   - Order $\mathcal{U}$ by $P(spec|\mathbf{x}_j)$ (expression 3)
   - $\mathcal{T}_{spec} \leftarrow$ most probable batch
   - $\mathcal{T}_{nspec} \leftarrow$ least probable batch
   - Train classifier using $\mathcal{T}_{spec}$ and $\mathcal{T}_{nspec}$

Rank	$\alpha = 0$	$\alpha = 1$	$\alpha = 5$	$\alpha = 100$	$\alpha = 500$
1	interactswith	suggest	suggest	suggest	suggest
2	TAFb	likely	likely	likely	likely
3	sexta	may	may	may	may
4	CRYs	might	might	These	These
5	DsRed	seems	seems	results	results
6	Cell-Nonautonomous	suggests	Taken	might	that
7	arva	probably	suggests	observations	be
8	inter-homologue	suggesting	probably	Taken	data
9	Mohanty	possibly	Together	findings	it
10	meld	suggested	suggesting	Our	Our
11	aDNA	Taken	possibly	seems	observations
12	Deer	unlikely	suggested	together	role
13	Borel	Together	findings	Together	most
14	substripe	physiology	observations	role	these
15	Failing	modulated	Given	that	together

Table 2: Features ranked by $P(spec|x_k)$ for varying α

- Compute *spec* recall/precision BEP (break-even point) on the test data

The batch size for each iteration is set to $0.001 * |\mathcal{U}|$. After each learning iteration, we compute the precision/recall BEP for the *spec* class using both classifiers trained on the current labelled data. We use BEP because it helps to mitigate against misleading results due to discrepancies in classification threshold placement. Disadvantageously, BEP does not measure a classifier's performance across the whole of the recall/precision spectrum (as can be obtained, for instance, from receiver-operating characteristic (ROC) curves), but for our purposes it provides a clear, abstracted overview of a classifier's accuracy given a particular training set.

10.2 Parameter Setting

The training and classification models we have presented require the setting of two parameters: the smoothing parameter α and the number of features per sample m. Analysis of the effect of varying α on feature ranking reveals that when $\alpha = 0$, low frequency terms with spurious class correlation dominate and as α increases, high frequency terms become increasingly dominant, eventually smoothing away genuine low-to-mid frequency correlations. This effect is illustrated in Table 2, and from this analysis we chose $\alpha = 5$ as an appropriate level of smoothing. We use $m = 5$ based on the intuition that five is a rough upper bound on the number of hedge cue features likely to occur in any one sentence.

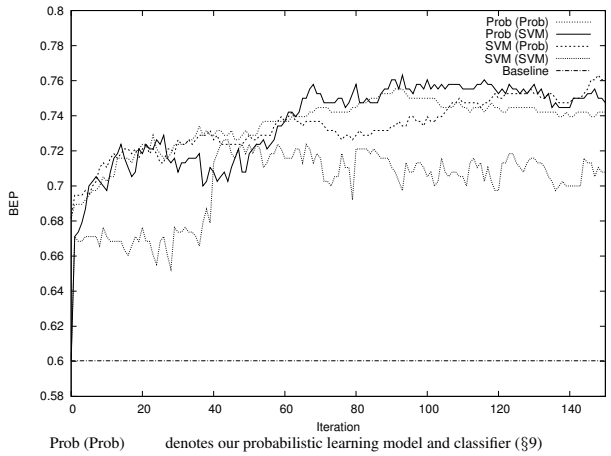
We use the linear kernel for SVM^{light} with the default setting for the regularization parameter C. We construct binary valued, L_2-normalised (unit length) input vectors to represent each sentence, as this resulted in better performance than using frequency-based weights and concords with our presence/absence feature estimates.

10.3 Seed Generation

The learning model we have presented requires a set of seeds for each class. To generate seeds for the *spec* class, we extracted all sentences from $\mathcal{U}$ containing either (or both) of the terms *suggest* or *likely*, as these are very good (though not perfect) hedge cues, yielding 6423 *spec* seeds. Generating seeds for *nspec* is much more difficult, as integrity requires the absence of hedge cues, and this cannot be done automatically. Thus, we used the following procedure to obtain a set of *nspec* seeds:

1. Create initial $\mathcal{S}_{nspec}$ by sampling randomly from $\mathcal{U}$.
2. Manually remove more 'obvious' speculative sentences using pattern matching
3. Iterate:
   - Order $\mathcal{S}_{nspec}$ by $P(spec|\mathbf{x}_j)$ using estimates from $\mathcal{S}_{spec}$ and current $\mathcal{S}_{nspec}$
   - Examine most probable sentences and remove speculative instances

We started with 8830 sentences and after a couple of hours work reduced this down to a (still potentially noisy) *nspec* seed set of 7541 sentences.



Prob (Prob)	denotes our probabilistic learning model and classifier (§9)
Prob (SVM)	denotes probabilistic learning model with SVM classifier
SVM (Prob)	denotes committee-based model (§10.4) with probabilistic classifier
SVM (SVM)	denotes committee-based model with SVM classifier
Baseline	denotes substring matching classifier of (Light et al., 2004)

Figure 1: Learning curves

10.4 Baselines

As a baseline classifier we use the substring matching technique of (Light et al., 2004), which labels a sentence as *spec* if it contains one or more of the following: *suggest*, *potential*, *likely*, *may*, *at least*, *in part*, *possibl*, *further investigation*, *unlikely*, *putative*, *insights*, *point toward*, *promise* and *propose*.

To provide a comparison for our learning model, we implement a more traditional self-training procedure in which at each iteration a committee of five SVMs is trained on randomly generated overlapping subsets of the training data and their cumulative confidence is used to select items for augmenting the labelled training data. For similar work see (Banko and Brill, 2001; Zhang, 2004).

10.5 Results

Figure 1 plots accuracy as a function of the training iteration. After 150 iterations, all of the weakly supervised learning models are significantly more accurate than the baseline according to a binomial sign test ($p < 0.01$), though there is clearly still much room for improvement. The baseline classifier achieves a BEP of 0.60 while both classifiers using our learning model reach approximately 0.76 BEP with little to tell between them. Interestingly, the combination of the SVM committee-based learning model with our classifier (denoted by 'SVM (Prob)'), performs competitively with both of the approaches that use our probabilistic learning model and significantly better than the SVM committee-based learning model with an SVM classifier, 'SVM (SVM)', according to a binomial sign test ($p < 0.01$) after 150 iterations. These results suggest that performance may be enhanced when the learning and classification tasks are carried out by different models. This is an interesting possibility, which we intend to explore further.

An important issue in incremental learning scenarios is identification of the optimum stopping point. Various methods have been investigated to address this problem, such as 'counter-training' (Yangarber, 2003) and committee agreement (Zhang, 2004); how such ideas can be adapted for this task is one of many avenues for future research.

10.6 Error Analysis

Some errors are due to the variety of hedge forms. For example, the learning models were unsuccessful in identifying assertive statements of knowledge paucity, eg: *There is no clear evidence for cytochrome c release during apoptosis in C elegans or Drosophila.* Whether it is possible to learn such examples without additional seed information is an open question. This example also highlights the potential benefit of an enriched sample representation, in this case one which accounts for the negation of the phrase 'clear evidence' which otherwise might suggest a strongly non-speculative assertion.

In many cases hedge classification is challenging even for a human annotator. For instance, distinguishing between a speculative assertion and one relating to a pattern of observed non-universal behaviour is often difficult. The following example was chosen by the learner as a *spec* sentence on the 150th training iteration: *Each component consists of a set of subcomponents that can be localized within a larger distributed neural system.* The sentence does not, in fact, contain a hedge but rather a statement of observed non-universal behaviour. However, an almost identical variant with 'could' instead of 'can' would be a strong speculative candidate. This highlights the similarity between many hedge and non-hedge instances, which makes such cases hard to learn in a weakly supervised manner.

11 Conclusions and Future Work

We have shown that weakly supervised ML is applicable to the problem of hedge classification and that a reasonable level of accuracy can be achieved. The work presented here has application in the wider academic community; in fact a key motivation in this study is to incorporate hedge classification into an interactive system for aiding curators in the construction and population of gene databases. We have presented our initial results on the task using a simple probabilistic model in the hope that this will encourage others to investigate alternative learning models and pursue new techniques for improving accuracy. Our next aim is to explore possibilities of introducing linguistically-motivated knowledge into the sample representation to help the learner identify key hedge-related sentential components, and also to consider hedge classification at the granularity of assertions rather than text sentences.

Acknowledgements

This work was partially supported by the FlySlip project, BBSRC Grant BBS/B/16291, and we thank Nikiforos Karamanis and Ruth Seal for thorough annotation and helpful discussion. The first author is supported by an University of Cambridge Millennium Scholarship.

References

Ron Artstein and Massimo Poesio. 2005. Kappa^3 = alpha (or beta). Technical report, University of Essex Department of Computer Science.

Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Meeting of the Association for Computational Linguistics*, pages 26–33.

Avrim Blum and Tom Mitchell. 1998. Combining labelled and unlabelled data with co-training. In *Proceedings of COLT' 98*, pages 92–100, New York, NY, USA. ACM Press.

Jinxiu Chen, Donghong Ji, Chew L. Tan, and Zhengyu Niu. 2006. Relation extraction using label propagation based semi-supervised learning. In *Proceedings of ACL'06*, pages 129–136.

M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in NLP and Very Large Corpora*.

George Hripcsak and Adam Rothschild. 2004. Agreement, the f-measure, and reliability in information retrieval. *J Am Med Inform Assoc.*, 12(3):296–298.

K. Hyland. 1994. Hedging in academic writing and eap textbooks. *English for Specific Purposes*, 13:239–256.

Thorsten Joachims. 1999. Making large-scale support vector machine learning practical. In A. Smola B. Schölkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA.

M. Light, X.Y. Qiu, and P. Srinivasan. 2004. The language of bioscience: Facts, speculations, and statements in between. In *Proceedings of BioLink 2004 Workshop on Linking Biological Literature, Ontologies and Databases: Tools for Users, Boston, May 2004*.

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

Vincent Ng and Claire Cardie. 2003. Weakly supervised natural language learning without redundant views. In *Proceedings of NAACL '03*, pages 94–101, Morristown, NJ, USA.

K. Nigam and R. Ghani. 2000. Understanding the behavior of co-training. In *Proceedings of KDD-2000 Workshop on Text Mining*.

Kamal Nigam, Andrew K. McCallum, Sebastian Thrun, and Tom M. Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134.

Ellen Riloff, Janyce Wiebe, and Theresa Wilson. 2003. Learning subjective nouns using extraction pattern bootstrapping. In *Seventh Conference on Natural Language Learning (CoNLL-03). ACL SIGNLL.*, pages 25–32.

Ben Wellner. 2005. Weakly supervised learning methods for improving the quality of gene name normalization data. In *Proceedings of the ACL-ISMB Workshop on Linking Biological Literature, Ontologies and Databases*, pages 1–8, Detroit, June. Association for Computational Linguistics.

Janyce Wiebe, Theresa Wilson, Rebecca Bruce, Matthew Bell, and Melanie Martin. 2004. Learning subjective language. *Comput. Linguist.*, 30(3):277–308.

Roman Yangarber. 2003. Counter-training in discovery of semantic patterns. In *Proceedings of ACL'03*, pages 343–350, Morristown, NJ, USA.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of ACL'95*, pages 189–196, Morristown, NJ, USA. ACL.

Zhu Zhang. 2004. Weakly-supervised relation classification for information extraction. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 581–588, New York, NY, USA. ACM Press.

Text Analysis for Automatic Image Annotation

Koen Deschacht and **Marie-Francine Moens**
Interdisciplinary Centre for Law & IT
Department of Computer Science
Katholieke Universiteit Leuven
Tiensestraat 41, 3000 Leuven, Belgium
{koen.deschacht,marie-france.moens}@law.kuleuven.ac.be

Abstract

We present a novel approach to automatically annotate images using associated text. We detect and classify all entities (persons and objects) in the text after which we determine the salience (the importance of an entity in a text) and visualness (the extent to which an entity can be perceived visually) of these entities. We combine these measures to compute the probability that an entity is present in the image. The suitability of our approach was successfully tested on 100 image-text pairs of Yahoo! News.

1 Introduction

Our society deals with a growing bulk of unstructured information such as text, images and video, a situation witnessed in many domains (news, biomedical information, intelligence information, business documents, etc.). This growth comes along with the demand for more effective tools to search and summarize this information. Moreover, there is the need to mine information from texts and images when they contribute to decision making by governments, businesses and other institutions. The capability to accurately recognize content in these sources would largely contribute to improved indexing, classification, filtering, mining and interrogation.

Algorithms and techniques for the disclosure of information from the different media have been developed for every medium independently during the last decennium, but only recently the interplay between these different media has become a topic of interest. One of the possible applications is to help analysis in one medium by employing information from another medium. In this paper we study text that is associated with an image, such as for instance image captions, video transcripts or surrounding text in a web page. We develop techniques that extract information from these texts to help with the difficult task of accurate object recognition in images. Although images and associated texts never contain precisely the same information, in many situations the associated text offers valuable information that helps to interpret the image.

The central objective of the CLASS project[1] is to develop advanced learning methods that allow images, video and associated text to be automatically analyzed and structured. In this paper we test the feasibility of automatically annotating images by using textual information in near-parallel image-text pairs, in which most of the content of the image corresponds to content of the text and vice versa. We will focus on entities such as persons and objects. We will hereby take into account the text's discourse structure and semantics, which allow a more fine-grained identification of what content might be present in the image, and will enrich our model with world knowledge that is not present in the text.

We will first discuss the corpus on which we apply and test our techniques in section 2, after which we outline what techniques we have developed: we start with a baseline system to annotate images with person names (section 3) and improve this by computing the importance of the persons in the text (section 4). We will then extend the model to include all

[1] http://class.inrialpes.fr/

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 1000–1007,
Prague, Czech Republic, June 2007.

Hiram Myers, of Edmond, Okla., walks across the fence, attempting to deliver what he called a 'people's indictment' of Halliburton CEO David Lesar, outside the site of the annual Halliburton shareholders meeting in Duncan, Okla., leading to his arrest, Wednesday, May 17, 2006.

Figure 1: Image-text pair with entity "Hiram Myers" appearing both in the text and in the image.

types of objects (section 5) and improve it by defining and computing the $visualness$ measure (section 6). Finally we will combine these different techniques in one probabilistic model in section 7.

2 The parallel corpus

We have created a parallel corpus consisting of 1700 image-text pairs, retrieved from the Yahoo! News website[2]. Every image has an accompanying text which describes the content of the image. This text will in general discuss one or more persons in the image, possibly one or more other objects, the location and the event for which the picture was taken. An example of an image-text pair is given in fig. 1. Not all persons or objects who are pictured in the images are necessarily described in the texts. The inverse is also true, i.e. content mentioned in the text may not be present in the image.

We have randomly selected 100 text-pairs from the corpus, and one annotator has labeled every image-text pair with the entities (i.e. persons and other objects) that appear both in the image and in the text. For example, the image-text pair shown in fig. 1 is annotated with one entity, "Hiram Myers", since this is the only entity that appears both in the text and in the image. On average these texts contain 15.04 entities, of which 2.58 appear in the image.

To build the appearance model of the text, we have combined different tools. We will evaluate every tool separately on 100 image-text pairs. This way we have a detailed view on the nature of the errors in the final model.

3 Automatically annotating person names

Given a text that is associated with an image, we want to compute a probabilistic *appearance model*, i.e. a collection of entities that are visible in the image. We will start with a model that holds the names of the persons that appear in the image, such as was done by (Satoh et al., 1999; Berg et al., 2004), and extend this model in section 5 to include all other objects.

3.1 Named Entity Recognition

A logical first step to detect person names is Named Entity Recognition (NER). We use the OpenNLP package[3], which detects noun phrase chunks in the sentences that represent persons, locations, organizations and dates. To improve the recognition of person names, we use a dictionary of names, which we have extracted from the Wikipedia[4] website. We have manually evaluated performance of NER on our test corpus and found that performance was satisfying: we obtained a precision of 93.37% and a recall of 97.69%. Precision is the percentage of identified person names by the system that corresponds to correct person names, and recall is the percentage of person names in the text that have been correctly identified by the system.

The texts contain a small number of noun phrase coreferents that are in the form of pronouns, we have resolved these using the LingPipe[5] package.

3.2 Baseline system

We want to annotate an image using the associated text. We try to find the names of persons which are

[2] http://news.yahoo.com/

[3] http://opennlp.sourceforge.net/

[4] http://en.wikipedia.org/

[5] http://www.alias-i.com/lingpipe/

both described in the text *and* visible in the image, and we want to do so by relying *only* on an analysis of the text. In some cases, such as the following example, the text states explicitly whether a person is (not) visible in the image:

> President Bush [...] with Danish Prime Minister Anders Fogh Rasmussen, not pictured, at Camp David [...].

Developing a system that could extract this information is not trivial, and even if we could do so, only a very small percentage of the texts in our corpus contain this kind of information. In the next section we will look into a method that is applicable to a wide range of (descriptive) texts and that does not rely on specific information within the text.

To evaluate the performance of this system, we will compare it with a simple baseline system. The baseline system assumes that all persons in the text are visible in the image, which results in a precision of 71.27% and a recall of 95.56%. The (low) precision can be explained by the fact that the texts often discuss people which are not present in the image.

4 Detection of the salience of a person

Not all persons discussed in a text are equally important. We would like to discover what persons are in the focus of a text and what persons are only mentioned briefly, because we presume that more important persons in the text have a larger probability of appearing in the image than less important persons. Because of the short lengths of the documents in our corpus, an analysis of lexical cohesion between terms in the text will not be sufficient for distinguishing between important and less important entities. We define a measure, *salience*, which is a number between 0 and 1 that represents the importance of an entity in a text. We present here a method for computing this score based on an in depth analysis of the discourse of the text and of the syntactic structure of the individual sentences.

4.1 Discourse segmentation

The discourse segmentation module, which we developed in earlier research, hierarchically and sequentially segments the discourse in different topics and subtopics resulting in a table of contents of a text (Moens, 2006). The table shows the main entities and the related subtopic entities in a tree-like structure that also indicates the segments (by means of character pointers) to which an entity applies. The algorithm detects patterns of thematic progression in texts and can thus recognize the main topic of a sentence (i.e., about whom or what the sentence speaks) and the hierarchical and sequential relationships between individual topics. A mixture model, taking into account different discourse features, is trained with the Expectation Maximization algorithm on an annotated DUC-2003 corpus. We use the resulting discourse segmentation to define the salience of individual entities that are recognized as topics of a sentence. We compute for each noun entity e_r in the discourse its salience ($Sal1$) in the discourse tree, which is proportional with the depth of the entity in the discourse tree -hereby assuming that deeper in this tree more detailed topics of a text are described- and normalize this value to be between zero and one. When an entity occurs in different subtrees, its maximum score is chosen.

4.2 Refinement with sentence parse information

Because not all entities of the text are captured in the discourse tree, we implement an additional refinement of the computation of the salience of an entity which is inspired by (Moens et al., 2006). The segmentation module already determines the main topic of a sentence. Since the syntactic structure is often indicative of the information distribution in a sentence, we can determine the relative importance of the other entities in a sentence by relying on the relationships between entities as signaled by the parse tree. When determining the salience of an entity, we take into account the level of the entity mention in the parse tree ($Sal2$), and the number of children for the entity in this structure ($Sal3$), where the normalized score is respectively inversely proportional with the depth of the parse tree where the entity occurs, and proportional with the number of children.

We combine the three salience values ($Sal1$, $Sal2$ and $Sal3$) by using a linear weighting. We have experimentally determined reasonable coefficients for these three values, which are respectively 0.8, 0.1 and 0.1. Eventually, we could learn these coefficients from a training corpus (e.g., with the

	Precision	Recall	F-measure
NER	71.27%	95.56%	81.65%
NER+DYN	97.66%	92.59%	95.06%

Table 1: Comparison of methods to predict what persons described in the text will appear in the image, using Named Entity Recognition (NER), and the salience measure with dynamic cut-off (DYN).

Expectation Maximization algorithm).

We do not separately evaluate our technology for salience detection as this technology was already extensively evaluated in the past (Moens, 2006).

4.3 Evaluating the improved system

The salience measure defines a ranking of all the persons in a text. We will use this ranking to improve our baseline system. We assume that it is possible to automatically determine the number of faces that are recognized in the image, which gives us an indication of a suitable cut-off value. This approach is reasonable since face detection (determine whether a face is present in the image) is significant easier than face recognition (determine which person is present in the image). In the improved model we assume that persons which are ranked higher than, or equal to, the cut-off value appear in the image. For example, if 4 faces appear in the image, we assume that only the 4 persons of which the names in the text have been assigned the highest salience appear in the image. We see from table 1 that the precision (97.66%) has improved drastically, while the recall remained high (92.59%). This confirms the hypothesis that determining the focus of a text helps in determining the persons that appear in the image.

5 Automatically annotating persons and objects

After having developed a reasonable successful system to detect what persons will appear in the image, we turn to a more difficult case : Detecting persons *and* all other objects that are described in the text.

5.1 Entity detection

We will first detect what words in the text refer to an entity. For this, we perform part-of-speech tagging (i.e., detecting the syntactic word class such as noun, verb, etc.). We take that every noun in the text represents an entity. We have used LTPOS (Mikheev, 1997), which performed the task almost errorless (precision of 98.144% and recall of 97.36% on the nouns in the test corpus). Person names which were segmented using the NER package are also marked as entities.

5.2 Baseline system

We want to detect the objects and the names of persons which are both visible in the image and described in the text. We start with a simple baseline system, in which we assume that every entity in the text appears in the image. As can be expected, this results in a high recall (91.08%), and a very low precision (15.62%). We see that the problem here is far more difficult compared to detecting only person names. This can be explained by the fact that many entities (such as for example *August*, *idea* and *history*) will never (or only indirectly) appear in an image. In the next section we will try to determine what types of entities are more likely to appear in the image.

6 Detection of the visualness of an entity

The assumption that every entity in the text appears in the image is rather crude. We will enrich our model with external world knowledge to find entities which are not likely to appear in an image. We define a measure called *visualness*, which is defined as the extent to which an entity can be perceived visually.

6.1 Entity classification

After we have performed entity detection, we want to classify every entity according to a certain semantic database. We use the WordNet (Fellbaum, 1998) database, which organizes English nouns, verbs, adjectives and adverbs in synsets. A synset is a collection of words that have a close meaning and that represent an underlying concept. An example of such a synset is “person, individual, someone, somebody, mortal, soul”. All these words refer to a hu-

man being. In order to correctly assign a noun in a text to its synset, i.e., to disambiguate the sense of this word, we use an efficient Word Sense Disambiguation (WSD) system that was developed by the authors and which is described in (Deschacht and Moens, 2006). Proper names are labeled by the Named Entity Recognizer, which recognizes persons, locations and organizations. These labels in turn allow us to assign the corresponding WordNet synset.

The combination of the WSD system and the NER package achieved a 75.97% accuracy in classifying the entities. Apart from errors that resulted from erroneous entity detection (32.32%), errors were mainly due to the WSD system (60.56%) and in a smaller amount to the NER package (8.12%).

6.2 WordNet similarity

We determine the visualness for every synset using a method that was inspired by Kamps and Marx (2002). Kamps and Marx use a distance measure defined on the adjectives of the WordNet database together with two seed adjectives to determine the emotive or affective meaning of any given adjective. They compute the relative distance of the adjective to the seed synsets "good" and "bad" and use this distance to define a measure of affective meaning.

We take a similar approach to determine the visualness of a given synset. We first define a similarity measure between synsets in the WordNet database. Then we select a set of seed synsets, i.e. synsets with a predefined visualness, and use the similarity of a given synset to the seed synsets to determine the visualness.

6.3 Distance measure

The WordNet database defines different relations between its synsets. An important relation for nouns is the hypernym/hyponym relation. A noun X is a hypernym of a noun Y if Y is a subtype or instance of X. For example, "bird" is a hypernym of "penguin" (and "penguin" is a hyponym of "bird"). A synset in WordNet can have one or more hypernyms. This relation organizes the synsets in a hierarchical tree (Hayes, 1999).

The similarity measure defined by Lin (1998) uses the hypernym/hyponym relation to compute a semantic similarity between two WordNet synsets S_1 and S_2. First it finds the most specific (lowest in the tree) synset S_p that is a parent of both S_1 and S_2. Then it computes the similarity of S_1 and S_2 as

$$sim(S_1, S_2) = \frac{2logP(S_p)}{logP(S_1) + logP(S_2)}$$

Here the probability $P(S_i)$ is the probability of labeling any word in a text with synset S_i or with one of the descendants of S_i in the WordNet hierarchy. We estimate these probabilities by counting the number of occurrences of a synset in the Semcor corpus (Fellbaum, 1998; Landes et al., 1998), where all noun chunks are labeled with their WordNet synset. The probability $P(S_i)$ is computed as

$$P(S_i) = \frac{C(S_i)}{\sum_{n=1}^{N} C(S_n)} + \sum_{k=1}^{K} P(S_k)$$

where $C(S_i)$ is the number of occurrences of S_i, N is the total number of synsets in WordNet and K is the number of children of S_i. The WordNet::Similarity package (Pedersen et al., 2004) implements this distance measure and was used by the authors.

6.4 Seed synsets

We have manually selected 25 seed synsets in WordNet, where we tried to cover the wide range of topics we were likely to encounter in the test corpus. We have set the visualness of these seed synsets to either 1 (visual) or 0 (not visual). We determine the visualness of all other synsets using these seed synsets. A synset that is close to a visual seed synset gets a high visualness and vice versa. We choose a linear weighting:

$$vis(s) = \sum_i vis(s_i) \frac{sim(s, s_i)}{C(s)}$$

where $vis(s)$ returns a number between 0 and 1 denoting the visualness of a synset s, s_i are the seed synsets, $sim(s, t)$ returns a number between 0 and 1 denoting the similarity between synsets s and t and $C(s)$ is constant given a synset s:

$$C(s) = \sum_i sim(s, s_i)$$

6.5 Evaluation of the visualness computation

To determine the visualness, we first assign the correct WordNet synset to every entity, after which we compute a visualness score for these synsets. Since these scores are floating point numbers, they are hard to evaluate manually. During evaluation, we make the simplifying assumption that all entities with a visualness below a certain threshold are not visual, and all entities above this threshold are visual. We choose this threshold to be 0.5. This results in an accuracy of 79.56%. Errors are mainly caused by erroneous entity detection and classification (63.10%) but also because of an incorrect assignment of the visualness (36.90%) by the method described above.

7 Creating an appearance model using salience and visualness

In the previous section we have created a method to calculate a visualness score for every entity, because we stated that removing the entities which can never be perceived visually will improve the performance of our baseline system. An experiment proves that this is exactly the case. If we assume that only the entities that have a visualness above a 0.5 threshold are visible and will appear in the image, we get a precision of 48.81% and a recall of 87.98%. We see from table 2 that this is already a significant improvement over the baseline system.

In section 4 we have seen that the salience measure helps in determining what persons are visible in the image. We have used the fact that face detection in images is relatively easily and can thus supply a cut-off value for the ranked person names. In the present state-of-the-art, we are not able to exploit a similar fact when detecting all types of entities. We will thus use the salience measure in a different way. We compute the salience of every entity, and we assume that only the entities with a salience score above a threshold of 0.5 will appear in the image. We see that this method drastically improves precision to 66.03%, but also lowers recall until 54.26%.

We now create a last model where we combine both the visualness and the salience measures. We want to calculate the probability of the occurrence of an entity e_{im} in the image, given a text t, $P(e_{im}|t)$. We assume that this probability is proportional with the degree of visualness and salience of e_{im} in t. In our framework, $P(e_{im}|t)$ is computed as the product of the salience of the entity e_{im} and its visualness score, as we assume both scores to be independent.

	Precision	Recall	F-measure
Ent	15.62%	91.08%	26.66%
Ent+Vis	48.81%	87.98%	62.78%
Ent+Sal	66.03%	54.26%	59.56%
Ent+Vis+Sal	70.56%	67.82%	69.39%

Table 2: Comparison of methods to predict the entities that appear in the image, using entity detection (Ent), and the visualness (Vis) and salience (Sal) measures.

Again, for evaluation sake, we choose a threshold of 0.4 to transform this continuous ranking into a binary classification. This results in a precision of 70.56% and a recall of 67.82%. This model is the best of the 4 models for entity annotation which have been evaluated.

8 Related Research

Using text that accompanies the image for annotating images and for training image recognition is not new. The earliest work (only on person names) is by Satoh (1999) and this research can be considered as the closest to our work. The authors make a distinction between proper names, common nouns and other words, and detect entities based on a thesaurus list of persons, social groups and other words, thus exploiting already simple semantics. Also a rudimentary approach to discourse analysis is followed by taking into account the position of words in a text. The results were not satisfactory: 752 words were extracted from video as candidates for being in the accompanying images, but only 94 were correct where 658 were false positives. Mori et al. (2000) learn textual descriptions of images from surrounding texts. These authors filter nouns and adjectives from the surrounding texts when they occur above a certain frequency and obtain a maximum hit rate of top 3 words that is situated between 30% and 40%. Other approaches consider both the textual and image features when building a content model of the image. For instance, some content is selected from the text (such as person names) and from the

image (such as faces) and both contribute in describing the content of a document. This approach was followed by Barnard (2003).

Westerveld (2000) combines image features and words from collateral text into one semantic space. This author uses Latent Semantic Indexing for representing the image/text pair content. Ayache et al. (2005) classify video data into different topical concepts. The results of these approaches are often disappointing. The methods here represent the text as a bag of words possibly augmented with a *tf* (term frequency) x *idf* (inverse document frequency) weight of the words (Amir et al., 2005). In exceptional cases, the hierarchical XML structure of a text document (which was manually annotated) is taken into account (Westerveld et al., 2005). The most interesting work here to mention is the work of Berg et al. (2004) who also process the nearly parallel image-text pairs found in the Yahoo! news corpus. They link faces in the image with names in the text (recognized with named entity recognition), but do not consider other objects. They consider pairs of person names (text) and faces (image) and use clustering with the Expectation Maximization algorithm to find all faces belonging to a certain person. In their model they consider the probability that an entity is pictured given the textual context (i.e., the part-of-speech tags immediately prior and after the name, the location of the name in the text and the distance to particular symbols such as "(R)"), which is learned with a probabilistic classifier in each step of the EM iteration. They obtained an accuracy of 84% on person face recognition.

In the CLASS project we work together with groups specialized in image recognition. In future work we will combine face and object recognition with text analysis techniques. We expect the recognition and disambiguation of faces to improve if many image-text pairs that treat the same person are used. On the other hand our approach is also valuable when there are few image-text pairs that picture a certain person or object. The approach of Berg et al. could be augmented with the typical features that we use, namely salience and visualness. In Deschacht et al. (2007) we have evaluated the ranking of persons and objects by the method we have described here and we have shown that this ranking correlates with the importance of persons and objects in the picture.

None of the above state-of-the-art approaches consider salience and visualness as discriminating factors in the entity recognition, although these aspects could advance the state-of-the-art.

9 Conclusion

Our society in the 21st century produces gigantic amounts of data, which are a mixture of different media. Our repositories contain texts interwoven with images, audio and video and we need automated ways to automatically index these data and to automatically find interrelationships between the various media contents. This is not an easy task. However, if we succeed in recognizing and aligning content in near-parallel image-text pairs, we might be able to use this acquired knowledge in indexing comparable image-text pairs (e.g., in video) by aligning content in these media.

In the experiment described above, we analyze the discourse and semantics of texts of near-parallel image-text pairs in order to compute the probability that an entity mentioned in the text is also present in the accompanying image. First, we have developed an approach for computing the salience of each entity mentioned in the text. Secondly, we have used the WordNet classification in order to detect the visualness of an entity, which is translated into a visualness probability. The combined salience and visualness provide a score that signals the probability that the entity is present in the accompanying image.

We extensively evaluated all the different modules of our system, pinpointing weak points that could be improved and exposing the potential of our work in cross-media exploitation of content.

We were able to detect the persons in the text that are also present in the image with a (evenly weighted) F-measure of more than 95%, and in addition were able to detect the entities that are present in the image with a F-measure of more than 69%. These results have been obtained by relying only on an analysis of the text and were substantially better than the baseline approach. Even if we can not resolve all ambiguity, keeping the most confident hypotheses generated by our textual hypotheses will greatly assist in analyzing images.

In the future we hope to extrinsically evaluate

the proposed technologies, e.g., by testing whether the recognized content in the text, improves image recognition, retrieval of multimedia sources, mining of these sources, and cross-media retrieval. In addition, we will investigate how we can build more refined appearance models that incorporate attributes and actions of entities.

Acknowledgments

The work reported in this paper was supported by the EU-IST project CLASS (Cognitive-Level Annotation using Latent Statistical Structure, IST-027978). We acknowledge the CLASS consortium partners for their valuable comments and we are especially grateful to Yves Gufflet from the INRIA research team (Grenoble, France) for collecting the Yahoo! News dataset.

References

Arnon Amir, Janne Argillander, Murray Campbell, Alexander Haubold, Giridharan Iyengar, Shahram Ebadollahi, Feng Kang, Milind R. Naphade, Apostol Natsev, John R. Smith, Jelena Tešió, and Timo Volkmer. 2005. IBM Research TRECVID-2005 Video Retrieval System. In *Proceedings of TRECVID 2005*, Gaithersburg, MD.

Stéphane Ayache, Gearges M. Qunot, Jrme Gensel, and Shin'Ichi Satoh. 2005. CLIPS-LRS-NII Experiments at TRECVID 2005. In *Proceedings of TRECVID 2005*, Gaithersburg, MD.

Kobus Barnard, Pinar Duygulu, Nando de Freitas, David Forsyth, David Blei, and Michael I. Jordan. 2003. Matching Words and Pictures. *Journal of Machine Learning Research*, 3(6):1107–1135.

Tamara L. Berg, Alexander C. Berg, Jaety Edwards, and D.A. Forsyth. 2004. Who's in the Picture? In *Neural Information Processing Systems*, pages 137–144.

Koen Deschacht and Marie-Francine Moens. 2006. Efficient Hierarchical Entity Classification Using Conditional Random Fields. In *Proceedings of the 2nd Workshop on Ontology Learning and Population*, pages 33–40, Sydney, July.

Koen Deschacht, Marie-Francine Moens, and W Robeyns. 2007. Cross-media entity recognition in nearly parallel visual and textual documents. In *Proceedings of the 8th RIAO Conference on Large-Scale Semantic Access to Content (Text, Image, Video and Sound)*. Cmu. (in press).

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

Brian Hayes. 1999. The Web of Words. *American Scientist*, 87(2):108–112, March-April.

Jaap Kamps and Maarten Marx. 2002. Words with Attitude. In *Proceedings of the 1st International Conference on Global WordNet*, pages 332–341, India.

Shari Landes, Claudia Leacock, and Randee I. Tengi. 1998. Building Semantic Concordances. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database*. The MIT Press.

Dekang Lin. 1998. An Information-Theoretic Definition of Similarity. In *Proc. 15th International Conf. on Machine Learning*.

Andrei Mikheev. 1997. Automatic Rule Induction for Unknown-Word Guessing. *Computational Linguistics*, 23(3):405–423.

Marie-Francine Moens, Patrick Jeuniaux, Roxana Angheluta, and Rudradeb Mitra. 2006. Measuring Aboutness of an Entity in a Text. In *Proceedings of HLT-NAACL 2006 TextGraphs: Graph-based Algorithms for Natural Language Processing*, East Stroudsburg. ACL.

Marie-Francine Moens. 2006. Using Patterns of Thematic Progression for Building a Table of Content of a Text. *Journal of Natural Language Engineering*, 12(3):1–28.

Yasuhide Mori, Hironobu Takahashi, and Ryuichi Oka. 2000. Automatic Word Assignment to Images Based on Image Division and Vector Quantization. In *RIAO-2000 Content-Based Multimedia Information Access*, Paris, April 12-14.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. WordNet::Similarity - Measuring the Relatedness of Concepts. In *The Proceedings of Fifth Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-04)*, Boston, May.

Shin'ichi Satoh, Yuichi Nakamura, and Takeo Kanade. 1999. Name-It: Naming and Detecting Faces in News Videos. *IEEE MultiMedia*, 6(1):22–35, January-March.

Thijs Westerveld, Jan C. van Gemert, Roberto Cornacchia, Djoerd Hiemstra, and Arjen de Vries. 2005. An Integrated Approach to Text and Image Retrieval. In *Proceedings of TRECVID 2005*, Gaithersburg, MD.

Thijs Westerveld. 2000. Image Retrieval: Content versus Context. In *Content-Based Multimedia Information Access, RIAO 2000 Conference Proceedings*, pages 276–284, April.

User Requirements Analysis for Meeting Information Retrieval Based on Query Elicitation

Vincenzo Pallotta
Department of Computer Science
University of Fribourg
Switzerland
Vincenzo.Pallotta@unifr.ch

Violeta Seretan
Language Technology Laboratory
University of Geneva
Switzerland
seretan@lettres.unige.ch

Marita Ailomaa
Artificial Intelligence Laboratory
Ecole Polytechnique Fédérale
de Lausanne (EPFL), Switzerland
Marita.Ailomaa@epfl.ch

Abstract

We present a user requirements study for Question Answering on meeting records that assesses the difficulty of users questions in terms of what type of knowledge is required in order to provide the correct answer. We grounded our work on the empirical analysis of elicited user queries. We found that the majority of elicited queries (around 60%) pertain to argumentative processes and outcomes. Our analysis also suggests that standard keyword-based Information Retrieval can only deal successfully with less than 20% of the queries, and that it must be complemented with other types of metadata and inference.

1 Introduction

Meeting records constitute a particularly important and rich source of information. Meetings are a frequent and sustained activity, in which multi-party dialogues take place that are goal-oriented and where participants perform a series of actions, usually aimed at reaching a common goal: they exchange information, raise issues, express opinions, make suggestions, propose solutions, provide arguments (pro or con), negotiate alternatives, and make decisions. As outcomes of the meeting, agreements on future action items are reached, tasks are assigned, conflicts are solved, etc. Meeting outcomes have a direct impact on the efficiency of organization and team performance, and the stored and indexed meeting records serve as reference for further processing (Post et al., 2004). They can also be used in future meetings in order to facilitate the decision-making process by accessing relevant information from previous meetings (Cremers et al., 2005), or in order to make the discussion more focused (Conklin, 2006).

Meetings constitute a substantial and important source of information that improves corporate organization and performance (Corrall, 1998; Romano and Nunamaker, 2001). Novel multimedia techniques have been dedicated to meeting *recording*, *structuring* and content *analysis* according to the metadata schema, and finally, to *accessing* the analyzed content via browsing, querying or filtering (Cremers et al., 2005; Tucker and Whittaker, 2004).

This paper focuses on *debate meetings* (Cugini et al., 1997) because of their particular richness in information concerning the decision-making process. We consider that the meeting content can be organized on three levels: (i) *factual level* (what happens: events, timeline, actions, dynamics); (ii) *thematic level* (what is said: topics discussed and details); (iii) *argumentative level* (which/how common goals are reached).

The information on the first two levels is explicit information that can be usually retrieved directly by searching the meeting records with appropriate IR techniques (i.e., TF-IDF). The third level, on the contrary, contains more abstract and tacit information pertaining to how the explicit information contributes to the rationale of the meeting, and it is not present as such in raw meeting data: whether or not the meeting goal was reached, what issues were debated, what proposals were made, what alternatives were discussed, what arguments were brought, what decisions were made, what task were assigned, etc.

The motivating scenario is the following: A user

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 1008–1015,
Prague, Czech Republic, June 2007.

needs information about a past meeting, either in quality of a participant who wants to recollect a discussion (since the memories of co-participants are often inconsistent, cf. Banerjee et al., 2005), or as a non-participant who missed that meeting. Instead of consulting the entire meeting-related information, which is usually heterogeneous and scaterred (audio-video recordings, notes, minutes, e-mails, handouts, etc.), the user asks natural language questions to a query engine which retrieves relevant information from the meeting records.

In this paper we assess the users' interest in retrieving argumentative information from meetings and what kind of knowledge is required for answering users' queries. Section 2 reviews previous user requirements studies for the meeting domain. Section 3 describes our user requirements study based on the analysis of elicited user queries, presents its main findings, and discusses the implications of these findings for the design of meeting retrieval systems. Section 4 concludes the paper and outlines some directions for future work.

2 Argumentative Information in Meeting Information Retrieval

Depending on the meeting browser type[1], different levels of meeting content become accessible for information retrieval. Audio and video browsers deal with factual and thematic information, while artifact browsers might also touch on deliberative information, as long as it is present, for instance, in the meeting minutes. In contrast, derived-data browsers aim to account for the argumentative information which is not explicitly present in the meeting content, but can be inferred from it. If minutes are likely to contain only the most salient deliberative facts, the derived-data browsers are much more useful, in that they offer access to the full meeting record, and thus to relevant details about the deliberative information sought.

2.1 Importance of Argumentative Structure

As shown by Rosemberg and Silince (1999), tracking argumentative information from meeting discussions is of central importance for building project memories since, in addition to the "strictly factual, technical information", these memories must also store relevant information about decision-making processes. In a business context, the information derived from meetings is useful for future business processes, as it can explain phenomena and past decisions and can support future actions by mining and assessment (Pallotta et al., 2004). The argumentative structure of meeting discussions, possibly visualized in form of argumentation diagrams or maps, can be helpful in meeting browsing. To our knowledge, there are at least three meeting browsers that have adopted argumentative structure: ARCHIVUS (Lisowska et al., 2004b), ViCoDe (Marchand-Maillet and Bruno, 2005), and the Twente-AMI JFerret browser (Rienks and Verbree, 2006).

2.2 Query Elicitation Studies

The users' interest in argumentation dimension of meetings has been highlighted by a series of recent studies that attempted to elicit the potential user questions about meetings (Lisowska et al., 2004a; Benerjee at al., 2005; Cremers et al., 2005).

The study of Lisowska et al. (2004a), part of the IM2 research project[2], was performed in a simulated environment in which users were asked to imagine themselves in a particular role from a series of scenarios. The participants were both IM2 members and non-IM2 members and produced about 300 retrospective queries on recorded meetings. Although this study has been criticized by Post et al. (2004), Cremers et al. (2005), and Banerjee et al. (2005) for being biased, artificial, obtrusive, and not conforming to strong HCI methodologies for survey research, it shed light on potential queries and classified them in two broad categories, that seem to correspond to our argumentative/non-argumentative distinction (Lisowska et al., 2004a: 994):

- “elements related to the interaction among participants: acceptance/rejection, agreement/disagreement; proposal, argumentation (for and against); assertions, statements; decisions; discussions, debates; reactions; questions; solutions”;

[1] (Tucker and Whittaker, 2004) identifies 4 types of meeting browsers: audio browsers, video browsers, artifacts browsers (that exploit meeting minutes or other meeting-related documents), and browsers that work with derived data (such as discourse and temporal structure information).

[2] http://www.im2.ch

- "concepts from the meeting domains: dates, times; documents; meeting index: current, previous, sets; participants; presentations, talks; projects; tasks, responsibilities; topics".

Unfortunately, the study does not provide precise information on the relative proportions of queries for the classification proposed, but simply suggests that overall more queries belong to the second category, while queries requiring understanding of the dialogue structure still comprise a sizeable proportion.

The survey conducted by Banerjee et al. (2005) concerned instead real, non-simulated interviews of busy professionals about actual situations, related either to meetings in which they previously participated, or to meetings they missed. More than half of the information sought by interviewees concerned, in both cases, the argumentative dimension of meetings.

For non-missed meetings, 15 out of the 26 instances (i.e., 57.7%) concerned argumentative aspects: what the decision was regarding a topic (7); what task someone was assigned (4); who made a particular decision (2); what was the participants' reaction to a particular topic (1); what the future plan is (1). The other instances (42.3%) relate to the thematic dimension, i.e., specifics of the discussion on a topic (11).

As for missed meetings, the argumentative instances were equally represented (18/36): decisions on a topic (7); what task was assigned to interviewee (4); whether a particular decision was made (3); what decisions were made (2); reasons for a decision (1); reactions to a topic (1). The thematic questions concern topics discussed, announcements made, and background of participants.

The study also showed that the recovery of information from meeting recordings is significantly faster when discourse annotations are available, such as the distinction between discussion, presentation, and briefing.

Another unobtrusive user requirements study was performed by Cremers et al. (2005) in a "semi-natural setting" related to the design of a meeting browser. The top 5 search interests highlighted by the 60 survey participants were: *decisions made*, participants/speakers, topics, agenda items, and *arguments for decision*. Of these, the ones shown in italics are argumentative. In fact, the authors acknowledge the necessity to include some "functional" categories as innovative search options. Interestingly, from the user interface evaluation presented in their paper, one can indirectly infer how salient the argumentative information is perceived by users: the icons that the authors intended for emotions, i.e., for a emotion-based search facility, were actually interpreted by users as referring to people's opinion: *What is person X's opinion?* – positive, negative, neutral.

3 User Requirements Analysis

The existing query elicitation experiments reported in Section 2 highlighted a series of question types that users typically would like to ask about meetings. It also revealed that the information sought can be classified into two broad categories: argumentative information (about the argumentative process and the outcome of debate meetings), and non-argumentative information (factual, i.e., about the meeting as a physical event, or thematic, i.e., about what has been said in terms of topics).

The study we present in this section is aimed at assessing how difficult it is to answer the questions that users typically ask about a meeting. Our goal is to provide insights into:

- how many queries can be answered using standard IR techniques on meeting artefacts only (e.g., minutes, written agenda, invitations);
- how many queries can be answered with IR on meeting recordings;
- what kind of additional information and inference is needed when IR does not apply or it is insufficient (e.g., information about the participants and the meeting dynamics, external information about the meeting's context such as the relation to a project, semantic interpretation of question terms and references, computation of durations, aggregation of results, etc).

Assessing the level of difficulty of a query based on the two above-mentioned categories might not provide insightful results, because these would be too general, thus less interpretable. Also, the complex queries requiring mixed information would escape observation because assigned to a too general class. We therefore considered it necessary to perform a separate analysis of each query instance, as this provides not only detailed, but also traceable information.

3.1 Data: Collecting User Queries

Our analysis is based on a heterogeneous collection of queries for meeting data. In general, an unbiased queries dataset is difficult to obtain, and the quality of a dataset can vary if the sample is made of too homogenous subjects (e.g., people belonging to the same group as members of the same project). In order to cope with this problem, our strategy was to use three different datasets collected in different settings:

- First, we considered the *IM2 dataset* collected by Lisowska et al. (2004a), the only set of user queries on meetings available to date. It comprises 270 questions (shortly described in Section 2) annotated with a label showing whether or not the query was produced by an IM2-member. These queries are introspective and not related to any particular recorded meeting.
- Second, we cross-validated this dataset with a large corpus of 294 natural language statements about existing meetings records. This dataset, called the *BET observations* (Wellner et al., 2005), was collected by subjects who were asked to watch several meeting recordings and to report what the meeting participants appeared to consider interesting. We use it as a 'validation' set for the IM2 queries: an IM2 query is considered as 'realistic' or 'empirically grounded' if there is a BET observation that represents a possible answer to the query. For instance, the query *Why was the proposal made by X not accepted?* matches the BET observation *Denis eliminated Silence of the Lambs as it was too violent.*
- Finally, we collected a new set of 'real' queries by conducting a survey of user requirements on meeting querying in a natural business setting. The survey involved 3 top managers from a company and produced 35 queries. We called this dataset *Manager Survey Set* (MS-Set).

The queries from the IM2-set (270 queries) and the MS-Set (35 queries) were analyzed by two different teams of two judges. Each team discussed each query, and classified it along the two main dimensions we are interested in:

- *query type*: the type of meeting content to which the query pertains;
- *query difficulty*: the type of information required to provide the answer.

3.2 Query Type Analysis

Each query was assigned exactly one of the following four possible categories (the one perceived as the most salient):

1. *factual*: the query pertains to the factual meeting content;
2. *thematic*: the query pertains to the thematic meeting content;
3. *process*: the query pertains to the argumentative meeting content, more precisely to the argumentative process;
4. *outcome*: the query pertains to the argumentative meeting content, more precisely to the outcome of the argumentative process.

Category	IM2-set (size:270)		MS-Set (size: 35)	
	Team1	Team2	Team1	Team2
Factual	24.8%	45.6%	20.0%	20.0%
Thematic	18.5%		20.0%	11.4%
Process	30.0%	32.6%	22.9%	28.6%
Outcome	26.7%	21.8%	37.1%	40.0%
Process+ Outcome	56.7%	54.4%	60.0%	68.6%

Table 1. Query classification according to the meeting content type.

Results from this classification task for both query sets are reported in Table 1. In both sets, the information most sought was argumentative: about 55% of the IM2-set queries are argumentative (process or outcome). This invalidates the initial estimation of Lisowska et al. (2004a:994) that the non-argumentative queries prevail, and confirms the figures obtained in (Banerjee et al., 2005), according to which 57.7% of the queries are argumentative. In our real managers survey, we obtained even higher percentages for the argumentative queries (60% or 68.6%, depending on the annotation team). The argumentative queries are followed by factual and thematic ones in both query sets, with a slight advantage for factual queries.

The inter-annotator agreement for this first classification is reported in Table 2. The proportion of queries on which annotators agree in classifying them as argumentative is significantly high. We only report here the agreement results for the individual argumentative categories (Process, Outcome) and both (Process & Outcome). There were 213 queries (in IM2-set) and 30 queries (in MS-

set) that were consistently annotated by the two teams on *both* categories. Within this set, a high percentage of queries were argumentative, that is, they were annotated as either Process or Outcome (label AA in the table).

Category	IM2-set (size: 270)		MS-set (size: 35)	
	ratio	kappa	ratio	kappa
Process	84.8%	82.9%	88.6%	87.8%
Outcome	90.7%	89.6%	91.4%	90.9%
Process & Outcome	78.9%	76.2%	85.7%	84.8%
AA	117/213 = **54.9%**		19/30 = **63.3%**	

Table 2. Inter-annotator agreement for query-type classification.

Furthermore, we provided a re-assessment of the proportion of argumentative queries with respect to query origin for the IM2-set (IM2 members vs. non-IM2 members): non-IM2 members issued 30.8% of agreed argumentative queries, a proportion that, while smaller compared to that of IM2 members (69.2%), is still non-negligible. This contrasts with the opinion expressed in (Lisowska et al., 2004a) that argumentative queries are almost exclusively produced by IM2 members.

Among the 90 agreed IM2 queries that were cross-validated with the BET-observation set, 28.9% were argumentative. We also noted that the ratio of BET statements that contain argumentative information is quite high (66.9%).

3.3 Query Difficulty Analysis

In order to assess the difficulty in answering a query, we used the following categories that the annotators could assign to each query, according to the type of information and techniques they judged necessary for answering it:

1. *Role of IR*: states the role of standard[3] Information Retrieval (in combination with Topic Extraction[4]) techniques in answering the query. Possible values:
    a. *Irrelevant* (IR techniques are not applicable). Example: *What decisions have been made?*
    b. *successful* (IR techniques are sufficient). Example: *Was the budget approved?*
    c. *insufficient* (IR techniques are necessary, but not sufficient alone since they require additional inference and information, such as argumentative, cross-meeting, external corporate/project knowledge). Example: *Who rejected the proposal made by X on issue Y?*
2. *Artefacts*: information such as agenda, minutes of previous meetings, e-mails, invitations and other documents related and available before the meeting. Example: *Who was invited to the meeting?*
3. *Recordings*: the meeting recordings (audio, visual, transcription). This is almost always true, except for queries where Artefacts or Metadata are sufficient, such as *What was the agenda?, Who was invited to the meeting?*).
4. *Metadata*: context knowledge kept in static metadata (e.g., speakers, place, time). Example: *Who were the participants at the meeting?*
5. *Dialogue Acts & Adjacency Pairs*: Example: *What was John's response to my comment on the last meeting?*
6. *Argumentation*: metadata (annotations) about the argumentative structure of the meeting content. Example: *Did everybody agree on the decisions, or were there differences of opinion?*
7. *Semantics*: semantic interpretation of terms in the query and reference resolution, including deictics (e.g., for *how long, usually, systematically, criticisms; this, about me, I*). Example: *What decisions got made <u>easily</u>?* The term requiring semantic interpretation is underlined.
8. *Inference*: inference (deriving information that is implicit), calculation, and aggregation (e.g., for 'command' queries asking for lists of things – participants, issues, proposals). Example: *What would be required from me?*

[3] By standard IR we mean techniques based on bag-of-word search and TF-IDF indexing.

[4] Topic extraction techniques are based on topic shift detection (Galley et al., 2003) and keyword extraction (van der Plas et al., 2004).

9. *Multiple meetings*: availability of multiple meeting records. Example: *Who usually attends the project meetings?*
10. *External*: related knowledge, not explicitly present in the meeting records (e.g., information about the corporation or the projects related to the meeting). Example: *Did somebody talk about me or about my work?*

Results of annotation reported on the two query sets are synthesized in Table 3: IR is sufficient for answering 14.4% of the IM2 queries, and 20% of the MS-set queries. In 50% and 25.7% of the cases, respectively, it simply cannot be applied (irrelevant). Finally, IR alone is not enough in 35.6% of the queries from the IM2-set, and in 54.3% of the MS-set; it has to be complemented with other techniques.

IR is:	IM2-set		MS-set	
	all queries	AA	all queries	AA
Sufficient	39/270 = 14.4%	1/117 = 0.8%	7/35 = 20.0%	1/19 = 5.3%
Irrelevant	135/270 = 50.0%	55/117 = 47.0%	9/35 = 25.7%	3/19 = 15.8%
Insufficient	96/270 = 35.6%	61/117 = **52.1%**	19/35 = 54.3%	15/19 = **78.9%**

Table 3. The role of IR (and topic extraction) in answering users' queries.

If we consider agreed argumentative queries (Section 3.2), IR is effective in an extremely low percentage of cases (0.8% for IM2-set and 5.3% for MS-Set). IR is insufficient in most of the cases (52.1% and 78.9%) and inapplicable in the rest of the cases (47% and 15.8%). Only one argumentative query from each set was judged as being answerable with IR alone: *What were the decisions to be made (open questions) regarding the topic t1? When is the NEXT MEETING planned? (e.g. to follow up on action items).*

Table 4 shows the number of queries in each set that require argumentative information in order to be answered, distributed according to the query types. As expected, no argumentation information is necessary for answering factual queries, but some thematic queries do need it, such as *What was decided about topic T?* (24% in the IM2-set and 42.9% in the M.S.-set).

Overall, the majority of queries in both sets require argumentation information in order to be answered (56.3% from IM2 queries, and 65.7% from MS queries).

Category	IM2-set, Annotation 1			MS-set, Annotation 1		
	total	Req. arg.	Ratio	Total	Req. arg.	Ratio
Factual	67	0	0%	7	0	0%
Thematic	50	12	24.0%	7	3	42.9%
Process	81	73	90.1%	8	7	87.5%
Outcome	72	67	93.1%	13	13	100%
All	270	152	**56.3%**	35	23	**65.7%**

Table 4. Queries requiring argumentative information.

We finally looked at what kind of information is needed in those cases where IR is perceived as insufficient or irrelevant. Table 5 lists the most frequent combinations of information types required for the IM2-set and the MS-set.

3.4 Summary of Findings

The analysis of the annotations obtained for the 305 queries (35 from the Manager Survey set, and 270 from the IM2-set) revealed that:

- The information most sought by users from meetings is argumentative (i.e., pertains to the argumentative process and its outcome). It constitutes more than half of the total queries, while factual and thematic information are similar in proportions (Table 1);
- There was no significant difference in this respect between the IM2-set and the MS-set (Table 1);
- The decision as to whether a query is argumentative or not is easy to draw, as suggested by the high inter-annotator agreement shown in Table 2;
- Standard IR and topic extraction techniques are perceived as insufficient in answering most of the queries. Only less than 20% of the whole query set can be answered with IR, and almost no argumentative question (Table 3).
- Argumentative information is needed in answering the majority of the queries (Table 4);
- When IR alone fails, the information types that are needed most are (in addition to recordings): Argumentation, Semantics, Inference, and Metadata (Table 5); see Section 3.3 for their description.

IR alone fails	IM2-set												
Information types	IR insufficient			96 cases		35.6%		IR irrelevant			135 cases		50%
Artefacts									x				
Recordings	x	x	x	x	x	x	x	x	x	x	x		
Meta-data			x		x			x		x		x	x
Dlg acts & Adj. pairs													
Argumentation	x	x	x	x	x	x	x	x	x		x		
Semantics	x	x	x	x	x			x	x	x	x	x	
Inference	x		x	x			x	x	x	x	x	x	
Multiple meetings				x								x	
External													
Cases	15	11	9	8	7	5	4	14	9	8	8	7	5
Ratio (%)	15.6	11.5	9.4	8.3	7.3	5.2	4.2	10.4	6.7	5.9	5.9	5.2	3.7

IR alone fails	MS-set					
Information types	IR insufficient		19 cases	54.3%	IR irrelevant	9 cases 54.3%
Artefacts					x	x
Recordings	x	x	x	x		
Meta-data					x	x
Dlg acts & Adj. pairs						
Argumentation	x	x	x	x		
Semantics	x		x	x	x	
Inference	x	x		x	x	
Multiple meetings						
External				x		
Cases	6	4	2	2	2	2
Ratio (%)	31.6	21	10.5	10.5	22.2	22.2

Table 5. Some of the most frequent combinations of information required for answering the queries in the IM2-Set and in the MS-set when IR alone fails.

3.5 Discussion

Searching relevant information through the recorded meeting dialogues poses important problems when using standard IR indexing techniques (Baeza-Yates and Ribeiro-Nieto, 2000), because users ask different types of queries for which a single retrieval strategy (e.g., keywords-based) is insufficient. This is the case when looking at answers that require some sort of entailment, such as inferring that a proposal has been rejected when a meeting participant says *Are you kidding?*.

Spoken-language information retrieval (Vinciarelli, 2004) and automatic dialogue-act extraction techniques (Stolke et al., 2000; Clark and Popescu-Belis, 2004; Ang et al., 2005) have been applied to meeting recordings and produced good results under the assumption that the user is interested in retrieving either topic-based or dialog act-based information. But this assumption is partially invalidated by our user query elicitation analysis, which showed that such information is only sought in a relatively small fraction of the users' queries. A particular problem for these approaches is that the topic looked for is usually not a query itself (*Was topic T mentioned?)*, but just a parameter in more structured questions (*What was decided about T?*). Moreover, the relevant participants' contributions (dialog acts) need to be retrieved in combination, not in isolation (*The reactions to the proposal made by X*).

4 Conclusion and Future Work

While most of the research community has neglected the importance of argumentative queries in meeting information retrieval, we provided evidence that this type of queries is actually very common. We quantified the proportion of queries involving the argumentative dimension of the meeting content by performing an in-depth analysis of queries collected in two different elicitation surveys. The analysis of the annotations obtained for the 305 queries (270 from the IM2-set, 35 from MS-set) was aimed at providing insights into different matters: what type of information is typically sought by users from meetings; how difficult it is, and what kind of information and techniques are needed in order to answer user queries.

This work represents an initial step towards a better understanding of user queries on the meeting domain. It could provide useful intuitions about

how to perform the automatic classification of answer types and, more importantly, the automatic extraction of argumentative features and their relations with other components of the query (e.g., topic, named entities, events).

In the future, we intend to better ground our first empirical findings by i) running the queries against a real IR system with indexed meeting transcripts and evaluate the quality of the obtained answers; ii) ask judges to manually rank the difficulty of each query, and iii) compare the two rankings. We would also like to see how frequent argumentative queries are in other domains (such as TV talk shows or political debates) in order to generalize our results.

Acknowledgements

We wish to thank Martin Rajman and Hatem Ghorbel for their constant and valuable feedback. This work has been partially supported by the Swiss National Science Foundation NCCR IM2 and by the SNSF grant no. 200021-116235.

References

Jeremy Ang, Yang Liu and Elizabeth Shriberg. 2005. Automatic Dialog Act Segmentation and Classification in Multiparty Meetings. In *Proceedings of IEEE ICASSP 2005*, Philadelphia, PA, USA.

Ricardo Baeza-Yates and Berthier Ribeiro-Nieto. 2000. *Modern Information Retrieval*. Addison Wesley.

Satanjeev Banerjee, Carolyn Rose and Alexander I. Rudnicky. 2005. The Necessity of a Meeting Recording and Playback System, and the Benefit of Topic-Level Annotations to Meeting Browsing. In *Proceedings of INTERACT 2005*, Rome, Italy.

Alexander Clark and Andrei Popescu-Belis. 2004. Multi-level Dialogue Act Tags. In *Proceedings of SIGDIAL'04*, pages 163–170. Cambridge, MA, USA.

Jeff Conklin. 2006. *Dialogue Mapping: Building Shared Understanding of Wicked Problems*. John Wiley & Sons.

Sheila Corrall. 1998. Knowledge management. Are we in the knowledge management business? *ARIADNE: the Web version*, 18.

Anita H.M Cremers, Bart Hilhorst and Arnold P.O.S Vermeeren. 2005. "What was discussed by whom, how, when and where?" personalized browsing of annotated multimedia meeting recordings. In *Proceedings of HCI 2005*, pages 1–10, Edinburgh, UK.

John Cugini, Laurie Damianos, Lynette Hirschman, Robyn Kozierok, Jeff Kurtz, Sharon Laskowski and Jean Scholtz. 1997. Methodology for evaluation of collaborative systems. Technical Report Rev. 3.0, The Evaluation Working Group of the DARPA Intelligent Collaboration and Visualization Program.

Michel Galley, Kathleen McKeown, Eric Fosler-Lussier and Hongyan Jing. 2003. Discourse Segmentation of Multi-Party Conversation. In *Proceedings of ACL 2003*, pages 562–569, Sapporo, Japan.

Agnes Lisowska, Andrei Popescu-Belis and Susan Armstrong. 2004a. User Query Analysis for the Specification and Evaluation of a Dialogue Processing and Retrieval System. In *Proceedings LREC 2004*, pages 993–996, Lisbon, Portugal.

Agnes Lisowska, Martin Rajman and Trung H. Bui. 2004b. ARCHIVUS: A System for Accesssing the Content of Recorded Multimodal Meetings. In *Proceedings of MLMI 2004*, Martigny, Switzerland.

Stéphane Marchand-Maillet and Eric Bruno. 2005. Collection Guiding: A new framework for handling large multimedia collections. In *Proceeding of AVIVDiLib05*, Cortona, Italy.

Vincenzo Pallotta, Hatem Ghorbel, Afzal Ballim, Agnes Lisowska and Stéphane Marchand-Maillet. 2004. Towards meeting information systems: Meeting knowledge management. In *Proceedings of ICEIS 2005*, pages 464–469, Porto, Portugal.

Lonneke van der Plaas, Vincenzo Pallotta, Martin Rajman and Hatem Ghorbel. 2004. Automatic keyword extraction from spoken text: A comparison between two lexical resources: the EDR and WordNet. In *Proceedings of the LREC 2004*, pages 2205–2208, Lisbon, Portugal.

Wilfried M. Post, Anita H.M. Cremers and Olivier Blanson Henkemans. 2004. A Research Environment for Meeting Behavior. In *Proceedings of the 3rd Workshop on Social Intelligence Design*, pages 159–165, University of Twente, Enschede, The Netherlands.

Rutger Rienks and Daan Verbree. 2006. About the Usefulness and Learnability of Argument–Diagrams from Real Discussions. In *Proceedings of MLMI 2006*, Washington DC, USA.

Nicholas C. Romano Jr. and Jay F. Nunamaker Jr. 2001. Meeting Analysis: Findings from Research and Practice. In *Proceedings of HICSS-34*, Maui, HI, IEEE Computer Society.

Duska Rosemberg and John A.A. Silince. 1999. Common ground in computer-supported collaborative argumentation. In *Proceedings of the CLSCL99*, Stanford, CA, USA.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema and Marie Meteer. 2000. Dialog Act Modeling for Automatic Tagging and Recognition of Conversational Speech. *Computational Linguistics*, 26(3):339–373.

Simon Tucker and Steve Whittaker. 2004. Accessing multimodal meeting data: systems, problems and possibilities. In *Proceedings of MLMI 2004*, Martigny, Switzerland.

Alessandro Vinciarelli. 2004. Noisy text categorization. In *Proceedings of ICPR 2004*, Cambridge, UK.

Pierre Wellner, Mike Flynn, Simon Tucker, Steve Whittaker. 2005. A Meeting Browser Evaluation Test. In *Proceedings of CHI 2005*, Portand, Oregon, USA.

Combining Multiple Knowledge Sources for Dialogue Segmentation in Multimedia Archives

Pei-Yun Hsueh
School of Informatics
University of Edinburgh
Edinburgh, UK EH8 9WL
p.hsueh@ed.ac.uk

Johanna D. Moore
School of Informatics
University of Edinburgh
Edinburgh, UK EH8 9WL
J.Moore@ed.ac.uk

Abstract

Automatic segmentation is important for making multimedia archives comprehensible, and for developing downstream information retrieval and extraction modules. In this study, we explore approaches that can segment multiparty conversational speech by integrating various knowledge sources (e.g., words, audio and video recordings, speaker intention and context). In particular, we evaluate the performance of a Maximum Entropy approach, and examine the effectiveness of multimodal features on the task of dialogue segmentation. We also provide a quantitative account of the effect of using ASR transcription as opposed to human transcripts.

1 Introduction

Recent advances in multimedia technologies have led to huge archives of audio-video recordings of multiparty conversations in a wide range of areas including clinical use, online video sharing services, and meeting capture and analysis. While it is straightforward to replay such recordings, finding information from the often lengthy archives is a more challenging task. Annotating implicit semantics to enhance browsing and searching of recorded conversational speech has therefore posed new challenges to the field of multimedia information retrieval.

One critical problem is how to divide unstructured conversational speech into a number of locally coherent segments. The problem is important for two reasons: First, empirical analysis has shown that annotating transcripts with semantic information (e.g., topics) enables users to browse and find information from multimedia archives more efficiently (Banerjee et al., 2005). Second, because the automatically generated segments make up for the lack of explicit orthographic cues (e.g., story and paragraph breaks) in conversational speech, dialogue segmentation is useful in many spoken language understanding tasks, including anaphora resolution (Grosz and Sidner, 1986), information retrieval (e.g., as input for the TREC Spoken Document Retrieval (SDR) task), and summarization (Zechner and Waibel, 2000).

This study therefore aims to explore whether a Maximum Entropy (MaxEnt) classifier can integrate multiple knowledge sources for segmenting recorded speech. In this paper, we first evaluate the effectiveness of features that have been proposed in previous work, with a focus on features that can be extracted automatically. Second, we examine other knowledge sources that have not been studied systematically in previous work, but which we expect to be good predictors of dialogue segments. In addition, as our ultimate goal is to develop an information retrieval module that can be operated in a fully automatic fashion, we also investigate the impact of automatic speech recognition (ASR) errors on the task of dialogue segmentation.

2 Previous Work

In previous work, the problem of automatic dialogue segmentation is often considered as similar to the problem of topic segmentation. Therefore, research has adopted techniques previously developed

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 1016–1023,
Prague, Czech Republic, June 2007.

to segment topics in text (Kozima, 1993; Hearst, 1997; Reynar, 1998) and in read speech (e.g., broadcast news) (Ponte and Croft, 1997; Allan et al., 1998). For example, lexical cohesion-based algorithms, such as LCSEG (Galley et al., 2003), or its word frequency-based predecessor TextTile (Hearst, 1997) capture topic shifts by modeling the similarity of word repetition in adjacent windows.

However, recent work has shown that LCSEG is less successful in identifying "agenda-based conversation segments" (e.g., *presentation*, *group discussion*) that are typically signalled by differences in group activity (Hsueh and Moore, 2006). This is not surprising since LCSEG considers only lexical cohesion. Previous work has shown that training a segmentation model with features that are extracted from knowledge sources other than words, such as speaker interaction (e.g., overlap rate, pause, and speaker change) (Galley et al., 2003), or participant behaviors, e.g., note taking cues (Banerjee and Rudnicky, 2006), can outperform LCSEG on similar tasks.

In many other fields of research, a variety of features have been identified as indicative of segment boundaries in different types of recorded speech. For example, Brown et al. (1980) have shown that a discourse segment often starts with relatively high pitched sounds and ends with sounds of pitch within a more compressed range. Passonneau and Litman (1993) identified that topic shifts often occur after a pause of relatively long duration. Other prosodic cues (e.g., pitch contour, energy) have been studied for their correlation with story segments in read speech (Tur et al., 2001; Levow, 2004; Christensen et al., 2005) and with theory-based discourse segments in spontaneous speech (e.g., direction-given monologue) (Hirschberg and Nakatani, 1996). In addition, head and hand/forearm movements are used to detect group-action based segments (McCowan et al., 2005; Al-Hames et al., 2005).

However, many other features that we expect to signal segment boundaries have not been studied systematically. For instance, speaker intention (i.e., dialogue act types) and conversational context (e.g., speaker role). In addition, although these features are expected to be complementary to one another, few of the previous studies have looked at the question how to use conditional approaches to model the correlation among features.

3 Methodology

3.1 Meeting Corpus

This study aims to explore approaches that can integrate multimodal information to discover implicit semantics from conversation archives. As our goal is to identify multimodal cues of segmentation in face-to-face conversation, we use the AMI meeting corpus (Carletta et al., 2006), which includes audio-video recordings, to test our approach. In particular, we are using 50 scenario-based meetings from the AMI corpus, in which participants are assigned to different roles and given specific tasks related to designing a remote control. On average, AMI meetings last 26 minutes, with over 4,700 words transpired. This corpus includes annotation for dialogue segmentation and topic labels. In the annotation process, annotators were given the freedom to subdivide a segment into subsegments to indicate when the group was discussing a subtopic. Annotators were also given a set of segment descriptions to be used as labels. Annotators were instructed to add a new label only if they could not find a match in the standard set. The set of segment descriptions can be divided to three categories: activity-based (e.g., presentation, discussion), issue-based (e.g., budget, usability), and functional segments (e.g., chitchat, opening, closing).

3.2 Preprocessing

The first step is to break a recorded meeting into minimal units, which can vary from sentence chunks to blocks of sentences. In this study, we use *spurts*, that is, consecutive speech with no pause longer than 0.5 seconds, as minimal units.

Then, to examine the difference between the set of features that are characteristic of segmentation at both coarse and fine levels of granularity, this study characterizes a dialogue as a sequence of segments that may be further divided into sub-segments. We take the theory-free dialogue segmentation annotations in the corpus and flatten the sub-segment structure and consider only two levels of segmentation: top-level segments and all sub-level segments.[1] We

[1]We take the spurts which the annotators choose as the beginning of a segment as the topic boundaries. On average,

observed that annotators tended to annotate activity-based segments only at the top level, whereas they often included sub-topics when segmenting issue-based segments. For example, a top-level *interface specialist presentation* segment can be divided into *agenda/equipment issues*, *user requirements*, *existing products*, and *look and usability* sub-level segments.

3.3 Intercoder Agreement

To measure intercoder agreement, we employ three different metrics: the kappa coefficient, PK, and WD. Kappa values measure how well a pair of annotators agree on where the segments break. PK is the probability that two spurts drawn randomly from a document are incorrectly identified as belonging to the same segment. WindowDiff (WD) calculates the error rate by moving a sliding window across the transcript counting the number of times the hypothesized and reference segment boundaries are different. While not uncontroversial, the use of these metrics is widespread. Table 1 shows the intercoder agreement of the top-level and sub-level segmentation respectively.

It is unclear whether the kappa values shown here indicate reliable intercoder agreement.[2] But given the low disagreement rate among codings in terms of the PK and WD scores, we will argue for the reliability of the annotation procedure used in this study. Also, to our knowledge the reported degree of agreement is the best in the field of meeting dialogue segmentation.[3]

Intercoder	Kappa	PK	WD
TOP	0.66	0.11	0.17
SUB	0.59	0.23	0.28

Table 1: Intercoder agreement of annotations at the top-level (TOP) and sub-level (SUB) segments.

the annotators marked 8.7 top-level segments and 14.6 sub-segments per meeting.

[2]In computational linguistics, kappa values over 0.67 point to reliable intercoder agreement. But Di Eugenio and Glass (2004) have found that this interpretation does not hold true for all tasks.

[3]For example, Gruenstein et al.(2005) report kappa (PK/WD) of 0.41(0.28/0.34) for determining the top-level and 0.45(0.27/0.35) for the sub-level segments in the ICSI meeting corpus.

3.4 Feature Extraction

As reported in Section 2, there is a wide range of features that are potentially characteristic of segment boundaries, and we expect to find some of them useful for automatic recognition of segment boundaries. The features we explore can be divided into the following five classes:

Conversational Features: We follow Galley et al. (2003) and extracted a set of conversational features, including the amount of overlapping speech, the amount of silence between speaker segments, speaker activity change, the number of cue words, and the predictions of LCSEG (i.e., the lexical cohesion statistics, the estimated posterior probability, the predicted class).

Lexical Features: We compile the list of words that occur more than once in the spurts that have been marked as a top-level or sub-segment boundary in the training set. Each spurt is then represented as a vector space of unigrams from this list.

Prosodic Features: We use the direct modelling approach proposed in Shriberg and Stolcke (2001) and include maximum F0 and energy of the spurt, mean F0 and energy of the spurt, pitch contour (i.e., slope) and energy at multiple points (e.g., the first and last 100 and 200 ms, the first and last quarter, the first and second half) of a spurt. We also include rate of speech, in-spurt silence, preceding and subsequent pauses, and duration. The rate of speech is calculated as both the number of words and the number of syllables spoken per second.

Motion Features: We measure the magnitude of relevant movements in the meeting room using methods that detect movements directly from video recordings in frames of 40 ms. Of special interest are the frontal shots as recorded by the close up cameras, the hand movements as recorded by the overview cameras, and shots of the areas of the room where presentations are made. We then average the magnitude of movements over the frames within a spurt as its feature value.

Contextual Features: These include dialogue act type[4] and speaker role (e.g., project manager, mar-

[4]In the annotations, each dialogue act is classified as one of 15 types, including acts about information exchange (e.g., Inform), acts about possible actions (e.g., Suggest), acts whose primary purpose is to smooth the social functioning (e.g., Be-positive), acts that are commenting on previous discussion (e.g.,

keting expert). As each spurt may consist of multiple dialogue acts, we represent each spurt as a vector of dialogue act types, wherein a component is 1 or 0 depending on whether the type occurs in the spurt.

3.5 Multimodal Integration Using Maximum Entropy Models

Previous work has used MaxEnt models for sentence and topic segmentation and shown that conditional approaches can yield competitive results on these tasks (Christensen et al., 2005; Hsueh and Moore, 2006). In this study, we also use a MaxEnt classifier[5] for dialogue segmentation under the typical supervised learning scheme, that is, to train the classifier to maximize the conditional likelihood over the training data and then to use the trained model to predict whether an unseen spurt in the test set is a segment boundary or not. Because continuous features have to be discretized for MaxEnt, we applied a histogram binning approach, which divides the value range into N intervals that contain an equal number of counts as specified in the histogram, to discretize the data.

4 Experimental Results

4.1 Probabilistic Models

The first question we want to address is whether the different types of characteristic multimodal features can be integrated, using the conditional MaxEnt model, to automatically detect segment boundaries. In this study, we use a set of 50 meetings, which consists of 17,977 spurts. Among these spurts, only 1.7% and 3.3% are top-level and subsegment boundaries. For our experiments we use 10-fold cross validation. The baseline is the result obtained by using LCSEG, an unsupervised approach exploiting only lexical cohesion statistics.

Table 2 shows the results obtained by using the same set of conversational (CONV) features used in previous work (Galley et al., 2003; Hsueh and Moore, 2006), and results obtained by using all the available features (ALL). The evaluation metrics PK and WD are conventional measures of error rates in segmentation (see Section 3.3). In Row 2, we see that using a MaxEnt classifier trained on the conversational features (CONV) alone improves over the LCSEG baseline by 15.3% for top-level segments and 6.8% for sub-level segements. Row 3 shows that combining additional knowledge sources, including lexical features (LX1) and the non-verbal features, prosody (PROS), motion (MOT), and context (CTXT), yields a further improvement (of 8.8% for top-level segmentation and 5.4% for sub-level segmentation) over the model trained on conversational features.

	TOP		SUB	
Error Rate	PK	WD	PK	WD
BASELINE(LCSEG)	0.40	0.49	0.40	0.47
MAXENT(CONV)	0.34	0.34	0.37	0.37
MAXENT(ALL)	0.30	0.33	0.34	0.36

Table 2: Compare the result of MaxEnt models trained with only conversational features (CONV) and with all available features (ALL).

4.2 Feature Effects

The second question we want to address is which knowledge sources (and combinations) are good predictors for segment boundaries. In this round of experiments, we evaluate the performance of different feature combinations. Table 3 further illustrates the impact of each feature class on the error rate metrics (PK/WD). In addition, as the PK and WD score do not reflect the magnitude of over- or underprediction, we also report on the average number of hypothesized segment boundaries (Hyp). The number of reference segments in the annotations is 8.7 at the top-level and 14.6 at the sub-level.

Rows 2-6 in Table 3 show the results of models trained with each individual feature class. We performed a one-way ANOVA to examine the effect of different feature classes. The ANOVA suggests a reliable effect of feature class ($F(5, 54) = 36.1$; $p < .001$). We performed post-hoc tests (Tukey HSD) to test for significant differences. Analysis shows that the model that is trained with lexical features alone (LX1) performs significantly worse than the LCSEG baseline ($p < .001$). This is due to the fact that cue words, such as *okay* and *now*, learned from the training data to signal seg-

Elicit-Assessment), and acts that allow complete segmentation (e.g., Stall).

[5]The parameters of the MaxEnt classifier are optimized using Limited-Memory Variable Metrics.

	TOP			SUB		
	Hyp	PK	WD	Hyp	PK	WD
BASELINE (LCSEG)	17.6	0.40	0.49	17.6	0.40	0.47
LX1	61.2	0.53	0.72	65.1	0.49	0.66
CONV	3.1	0.34	0.34	2.9	0.37	0.37
PROS	2.3	0.35	0.35	2.5	0.37	0.37
MOT	96.2	0.36	0.40	96.2	0.38	0.41
CTXT	2.6	0.34	0.34	2.2	0.37	0.37
ALL	7.7	0.29	0.33	7.6	0.35	0.38

Table 3: Effects of individual feature classes and their combination on detecting segment boundaries.

ment boundaries, are often used for non-discourse purposes, such as making a semantic contribution to an utterance.[6] Thus, we hypothesize that these ambiguous cue words have led the LX1 model to overpredict. Row 7 further shows that when all available features (including LX1) are used, the combined model (ALL) yields performance that is significantly better than that obtained with individual feature classes ($F(5, 54) = 32.2$; $p < .001$).

	TOP			SUB		
	Hyp	PK	WD	Hyp	PK	WD
ALL	7.7	0.29	0.33	7.6	0.35	0.38
ALL-LX1	3.9	0.35	0.35	3.5	0.37	0.38
ALL-CONV	6.6	0.30	0.34	6.8	0.35	0.37
ALL-PROS	5.6	0.29	0.31	7.4	0.33	0.35
ALL-MOTION	7.5	0.30	0.35	7.3	0.35	0.37
ALL-CTXT	7.2	0.29	0.33	6.7	0.36	0.38

Table 4: Performance change of taking out each individual feature class from the ALL model.

Table 4 illustrates the error rate change (i.e., increased or decreased PK and WD score)[7] that is incurred by leaving out one feature class from the ALL model. Results show that CONV, PROS, MOTION and CTXT can be taken out from the ALL model individually without increasing the error rate significantly.[8] Morevoer, the combined models always perform better than the LX1 model ($p < .01$), cf. Table 3.

This suggests that the non-lexical feature classes are complementary to LX1, and thus it is essential to incorporate some, but not necessarily all, of the non-lexical classes into the model.

	TOP			SUB		
	Hyp	PK	WD	Hyp	PK	WD
LX1	61.2	0.53	0.72	65.1	0.49	0.66
MOT	96.2	0.36	0.40	96.2	0.38	0.41
LX1+CONV	5.3	0.27	0.30	6.9	0.32	0.35
LX1+PROS	6.2	0.30	0.33	7.3	0.36	0.38
LX1+MOT	20.2	0.39	0.49	24.8	0.39	0.47
LX1+CTXT	6.3	0.28	0.31	7.2	0.33	0.35
MOT+PROS	62.0	0.34	0.34	62.1	0.37	0.37
MOT+CTXT	2.7	0.33	0.33	2.3	0.37	0.37

Table 5: Effects of combining complementary features on detecting segment boundaries.

Table 5 further illustrates the performance of different feature combinations on detecting segment boundaries. By subtracting the PK or WD score in Row 1, the LX1 model, from that in Rows 3-6, we can tell how essential each of the non-lexical classes is to be combined with LX1 into one model. Results show that CONV is the most essential, followed by CTXT, PROS and MOT. The advantage of incorporating the non-lexical feature classes is also shown in the noticeably reduced number of overpredictions as compared to that of the LX1 model.

To analyze whether there is a significant interaction between feature classes, we performed another round of ANOVA tests to examine the effect of LX1 and each of the non-lexical feature classes on detecting segment boundaries. This analysis shows that there is a significant interaction effect on detecting both top-level and sub-level segment boundaries ($p < .01$), suggesting that the performance of LX1 is significantly improved when combined with any non-lexical feature class. Also, among the non-lexical feature classes, combining prosodic features significantly improves the performance of the model in which the motion features are combined to detect top-level segment boundaries ($p < .05$).

[6] Hirschberg and Litman (1987) have proposed to discriminate the different uses intonationally.

[7] Note that the increase in error rate indicates performance degradation, and vice versa.

[8] Sign tests were used to test for significant differences between means in each fold of cross validation.

4.3 Degradation Using ASR

The third question we want to address here is whether using the output of ASR will cause significant degradation to the performance of the segmentation approaches. The ASR transcripts used in this experiment are obtained using standard technology including HMM based acoustic modeling and N-gram based language models (Hain et al., 2005). The average word error rates (WER) are 39.1%. We also applied a word alignment algorithm to ensure that the number of words in the ASR transcripts is the same as that in the human-produced transcripts. In this way we can compare the PK and WD metrics obtained on the ASR outputs directly with that on the human transcripts.

In this study, we again use a set of 50 meetings and 10-fold cross validation. We compare the performance of the reference models, which are trained on human transcripts and tested on human transcripts, with that of the ASR models, which are trained on ASR transcripts and tested on ASR transcripts. Table 6 shows that despite the word recognition errors, none of the LCSEG, the MaxEnt models trained with conversational features, and the MaxEnt models trained with all available features perform significantly worse on ASR transcripts than on reference transcripts. One possible explanation for this, which we have observed in our corpus, is that the ASR system is likely to mis-recognize different occurences of words in the same way, and thus the lexical cohesion statistic, which captures the similarity of word repetition between two adjacency windows, is also likely to remain unchanged. In addition, when the models are trained with other features that are not affected by the recognition errors, such as pause and overlap, the negative impacts of recognition errors are further reduced to an insignificant level.

	TOP		SUB	
Error Rate	PK	WD	PK	WD
LCSEG(REF)	0.45	0.57	0.42	0.47
LCSEG(ASR)	0.45	0.58	0.40	0.47
MAXENT-CONV(REF)	0.34	0.34	0.37	0.37
MAXENT-CONV(ASR)	0.34	0.33	0.38	0.38
MAXENT-ALL(REF)	0.30	0.33	0.34	0.36
MAXENT-ALL(ASR)	0.31	0.34	0.34	0.37

Table 6: Effects of word recognition errors on detecting segments boundaries.

5 Discussion

The results in Section 4 show the benefits of including additional knowledge sources for recognizing segment boundaries. The next question to be addressed is what features in these sources are most useful for recognition. To provide a qualitative account of the segmentation cues, we performed an analysis to determine whether each proposed feature discriminates the class of segment boundaries. Previous work has identified statistical measures (e.g., Log Likelihood ratio) that are useful for determining the statistical association strength (relevance) of the occurrence of an n-gram feature to target class (Hsueh and Moore, 2006). Here we extend that study to calculate the LogLikelihood relevance of all of the features used in the experiments, and use the statistics to rank the features.

Our analysis shows that people do speak and behave differently near segment boundaries. Some of the identified segmentation cues match previous findings. For example, a segment is likely to start with higher pitched sounds (Brown et al., 1980; Ayers, 1994) and a lower rate of speech (Lehiste, 1980). Also, interlocutors pause longer than usual to make sure that everyone is ready to move on to a new discussion (Brown et al., 1980; Passonneau and Litman, 1993) and use some conventional expressions (e.g., *now*, *okay*, *let's*, *um*, *so*).

Our analysis also identified segmentation cues that have not been mentioned in previous research. For example, interlocutors do not move around a lot when a new discussion is brought up; interlocutors mention agenda items (e.g., *presentation*, *meeting*) or content words more often when initiating a new discussion. Also, from the analysis of current dialogue act types and their immediate contexts, we also observe that at segment boundaries interlocutors do the following more often than usual: start speaking before they are ready (*Stall*), give information (*Inform*), elicit an assessment of what has been said so far (*Elicit-assessment*), or act to smooth social functioning and make the group happier (*Be-positive*).

6 Conclusions and Future Work

This study explores the use of features from multiple knowledge sources (i.e., words, prosody, motion, interaction cues, speaker intention and role) for developing an automatic segmentation component in spontaneous, multiparty conversational speech. In particular, we addressed the following questions: (1) Can a MaxEnt classifier integrate the potentially characteristic multimodal features for automatic dialogue segmentation? (2) What are the most discriminative knowledge sources for detecting segment boundaries? (3) Does the use of ASR transcription significantly degrade the performance of a segmentation model?

First of all, our results show that a well performing MaxEnt model can be trained with available knowledge sources. Our results improve on previous work, which uses only conversational features, by 8.8% for top-level segmentation and 5.4% for sub-level segmentation. Analysis of the effectiveness of the various features shows that lexical features (i.e., cue words) are the most essential feature class to be combined into the segmentation model. However, lexical features must be combined with other features, in particular, conversational features (i.e., lexical cohesion, overlap, pause, speaker change), to train well performing models.

In addition, many of the non-lexical feature classes, including those that have been identified as indicative of segment boundaries in previous work (e.g., prosody) and those that we hypothesized as good predictors of segment boundaries (e.g., motion, context), are not beneficial for recognizing boundaries when used in isolation. However, these non-lexical features are useful when combined with lexical features, as the presence of the non-lexical features can balance the tendency of models trained with lexical cues alone to overpredict.

Experiments also show that it is possible to segment conversational speech directly on the ASR outputs. These results encouragingly show that we can segment conversational speech using features extracted from different knowledge sources, and in turn, facilitate the development of a fully automatic segmentation component for multimedia archives.

With the segmentation models developed and discriminative knowledge sources identified, a remaining question is whether it is possible to automatically select the discriminative features for recognition. This is particularly important for prosodic features, because the direct modelling approach we adopted resulted in a large number of features. We expect that by applying feature selection methods we can further improve the performance of automatic segmentation models. In the field of machine learning and pattern analysis, many methods and selection criteria have been proposed. Our next step will be to examine the effectiveness of these methods for the task of automatic segmentation. Also, we will further explore how to choose the best performing ensemble of knowledge sources so as to facilitate automatic selection of knowledge sources to be included.

Acknowledgement

This work was supported by the EU 6th FWP IST Integrated Project AMI (Augmented Multi-party Interaction, FP6-506811). Our special thanks to Wessel Kraaij, Stephan Raaijmakers, Steve Renals, Gabriel Murray, Jean Carletta, and the anonymous reviewers for valuable comments. Thanks also to the AMI ASR group for producing the ASR transcriptions, and to our research partners in TNO for generating motion features.

References

M. Al-Hames, A. Dielmann, D. GaticaPerez, S. Reiter, S. Renals, and D. Zhang. 2005. Multimodal integration for meeting group action segmentation and recognition. In *Proc. of MLMI 2005*.

J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. 1998. Topic detection and tracking pilot study: Final report. In *Proc. of the DARPA Broadcast News Transcription and Understanding Workshop*.

G. M. Ayers. 1994. Discourse functions of pitch range in spontaneous and read speech. In Jennifer J. Venditti, editor, *OSU Working Papers in Linguistics*, volume 44, pages 1–49.

S. Banerjee and A. Rudnicky. 2006. Segmenting meetings into agenda items by extracting implicit supervision from human note-taking. In *Proc. of IUI 2006*.

S. Banerjee, C. Rose, and A. I. Rudnicky. 2005. The necessity of a meeting recording and playback system, and the benefit of topic-level annotations to meeting

browsing. In *Proc. of the Tenth International Conference on Human-Computer Interaction*.

G. Brown, K. L. Currie, and J. Kenworthe. 1980. *Questions of Intonation*. University Park Press.

J. Carletta et al. 2006. The AMI meeting corpus: A preannouncement. In Steve Renals and Samy Bengio, editors, *Springer-Verlag Lecture Notes in Computer Science*, volume 3869. Springer-Verlag.

H. Christensen, B. Kolluru, Y. Gotoh, and S. Renals. 2005. Maximum entropy segmentation of broadcast news. In *Proc. of ICASP*, Philadelphia USA.

B. Di Eugenio and M. G. Glass. 2004. The kappa statistic: A second look. *Computational Linguistics*, 30(1):95–101.

M. Galley, K. McKeown, E. Fosler-Lussier, and H. Jing. 2003. Discourse segmentation of multi-party conversation. In *Proc. of ACL 2003*.

B. Grosz and C. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3).

A. Gruenstein, J. Niekrasz, and M. Purver. 2005. Meeting structure annotation: Data and tools. In *Proc. of the SIGdial Workshop on Discourse and Dialogue*.

T. Hain, J. Dines, G. Garau, M. Karafiat, D. Moore, V. Wan, R. Ordelman, and S. Renals. 2005. Transcription of conference room meetings: An investigation. In *Proc. of Interspeech 2005*.

M. Hearst. 1997. TextTiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 25(3):527–571.

J. Hirschberg and D. Litman. 1987. Now let's talk about now: identifying cue phrases intonationally. In *Proc. of ACL 1987*.

J. Hirschberg and C. H. Nakatani. 1996. A prosodic analysis of discourse segments in direction-giving monologues. In *Proc. of ACL 1996*.

P. Hsueh and J.D. Moore. 2006. Automatic topic segmentation and lablelling in multiparty dialogue. In *the first IEEE/ACM workshop on Spoken Language Technology (SLT) 2006*.

H. Kozima. 1993. Text segmentation based on similarity between words. In *Proc. of ACL 1993*.

I. Lehiste. 1980. Phonetic characteristics of discourse. In *the Meeting of the Committee on Speech Research, Acoustical Society of Japan*.

G. Levow. 2004. Prosody-based topic segmentation for mandarin broadcast news. In *Proc. of HLT 2004*.

I. McCowan, D. Gatica-Perez, S. Bengio, G. Lathoud, M. Barnard, and D. Zhang. 2005. Automatic analysis of multimodal group actions in meetings. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(3):305–317.

R. Passonneau and D. Litman. 1993. Intention-based segmentation: Human reliability and correlation with linguistic cues. In *Proc. of ACL 1993*.

J. Ponte and W. Croft. 1997. Text segmentation by topic. In *Proc. of the Conference on Research and Advanced Technology for Digital Libraries 1997*.

J. Reynar. 1998. *Topic Segmentation: Algorithms and Applications*. Ph.D. thesis, UPenn, PA USA.

E. Shriberg and A. Stolcke. 2001. Direct modeling of prosody: An overview of applications in automatic speech processing. In *Proc. International Conference on Speech Prosody 2004*.

G. Tur, D. Hakkani-Tur, A. Stolcke, and E. Shriberg. 2001. Integrating prosodic and lexical cues for automatic topic segmentation. *Computational Linguistics*, 27(1):31–57.

K. Zechner and A. Waibel. 2000. DIASUMM: Flexible summarization of spontaneous dialogues in unrestricted domains. In *Proc. of COLING-2000*.

Topic Analysis for Psychiatric Document Retrieval

Liang-Chih Yu*‡, Chung-Hsien Wu*, Chin-Yew Lin†, Eduard Hovy‡ and Chia-Ling Lin*
*Department of CSIE, National Cheng Kung University, Tainan, Taiwan, R.O.C.
†Microsoft Research Asia, Beijing, China
‡Information Sciences Institute, University of Southern California, Marina del Rey, CA, USA
liangchi@isi.edu,{chwu,totalwhite}@csie.ncku.edu.tw,cyl@microsoft.com,hovy@isi.edu

Abstract

Psychiatric document retrieval attempts to help people to efficiently and effectively locate the consultation documents relevant to their depressive problems. Individuals can understand how to alleviate their symptoms according to recommendations in the relevant documents. This work proposes the use of high-level topic information extracted from consultation documents to improve the precision of retrieval results. The topic information adopted herein includes *negative life events*, *depressive symptoms* and *semantic relations* between symptoms, which are beneficial for better understanding of users' queries. Experimental results show that the proposed approach achieves higher precision than the word-based retrieval models, namely the vector space model (VSM) and Okapi model, adopting word-level information alone.

1 Introduction

Individuals may suffer from negative or stressful life events, such as death of a family member, argument with a spouse and loss of a job. Such events play an important role in triggering depressive symptoms, such as depressed moods, suicide attempts and anxiety. Individuals under these circumstances can consult health professionals using message boards and other services. Health professionals respond with suggestions as soon as possible. However, the response time is generally several days, depending on both the processing time required by health professionals and the number of problems to be processed. Such a long response time is unacceptable, especially for patients suffering from psychiatric emergencies such as suicide attempts. A potential solution considers the problems that have been processed and the corresponding suggestions, called consultation documents, as the psychiatry web resources. These resources generally contain thousands of consultation documents (problem-response pairs), making them a useful information resource for mental health care and prevention. By referring to the relevant documents, individuals can become aware that they are not alone because many people have suffered from the same or similar problems. Additionally, they can understand how to alleviate their symptoms according to recommendations. However, browsing and searching all consultation documents to identify the relevant documents is time consuming and tends to become overwhelming. Individuals need to be able to retrieve the relevant consultation documents efficiently and effectively. Therefore, this work presents a novel mechanism to automatically retrieve the relevant consultation documents with respect to users' problems.

Traditional information retrieval systems represent queries and documents using a bag-of-words approach. Retrieval models, such as the *vector space model (VSM)* (Baeza-Yates and Ribeiro-Neto, 1999) and *Okapi model* (Robertson et al., 1995; Robertson et al., 1996; Okabe et al., 2005), are then adopted to estimate the relevance between queries and documents. The VSM represents each query and document as a vector of words, and adopts the *cosine measure* to estimate their relevance. The Okapi model, which has been used on the Text REtrieval Conference (TREC) collections, developed a family of word-weighting functions

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 1024–1031,
Prague, Czech Republic, June 2007.

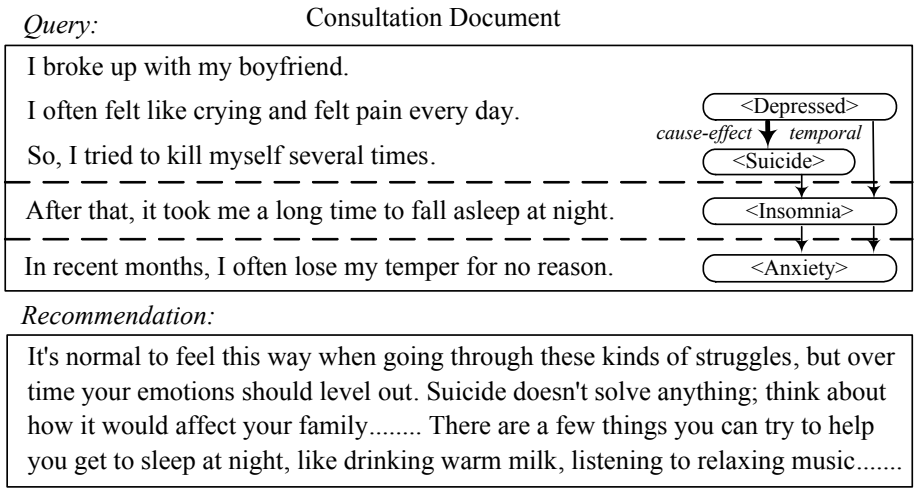


Figure 1. Example of a consultation document. The bold arrowed lines denote cause-effect relations; arrowed lines denote temporal relations; dashed lines denote temporal boundaries, and angle brackets denote depressive symptoms

for relevance estimation. These functions consider word frequencies and document lengths for word weighting. Both the VSM and Okapi models estimate the relevance by matching the words in a query with the words in a document. Additionally, query words can further be expanded by the concept hierarchy within general-purpose ontologies such as WordNet (Fellbaum, 1998), or automatically constructed ontologies (Yeh et al., 2004).

However, such word-based approaches only consider the word-level information in queries and documents, ignoring the high-level topic information that can help improve understanding of users' queries. Consider the example consultation document in Figure 1. A consultation document comprises two parts: the *query* part and *recommendation* part. The query part is a natural language text, containing rich *topic information* related to users' depressive problems. The topic information includes *negative life events*, *depressive symptoms*, and *semantic relations* between symptoms. As indicated in Figure 1, the subject suffered from a love-related event, and several depressive symptoms, such as <Depressed>, <Suicide>, <Insomnia> and <Anxiety>. Moreover, there is a cause-effect relation holding between <Depressed> and <Suicide>, and a temporal relation holding between <Depressed> and <Insomnia>. Different topics may lead to different suggestions decided by experts. Therefore, an ideal retrieval system for consultation documents should consider such topic information so as to improve the retrieval precision.

Natural language processing (NLP) techniques can be used to extract more precise information from natural language texts (Wu et al., 2005a; Wu et al., 2005b; Wu et al., 2006; Yu et al., 2007). This work adopts the methodology presented in (Wu et al. 2005a) to extract depressive symptoms and their relations, and adopts the pattern-based method presented in (Yu et al., 2007) to extract negative life events from both queries and consultation documents. This work also proposes a retrieval model to calculate the similarity between a query and a document by combining the similarities of the extracted topic information.

The rest of this work is organized as follows. Section 2 briefly describes the extraction of topic information. Section 3 presents the retrieval model. Section 4 summarizes the experimental results. Conclusions are finally drawn in Section 5.

2 Framework of Consultation Document Retrieval

Figure 2 shows the framework of consultation document retrieval. The retrieval process begins with receiving a user's query about his depressive problems in natural language. The example query is shown in Figure 1. The topic information is then extracted from the query, as shown in the center of Figure 2. The extracted topic information is repre-

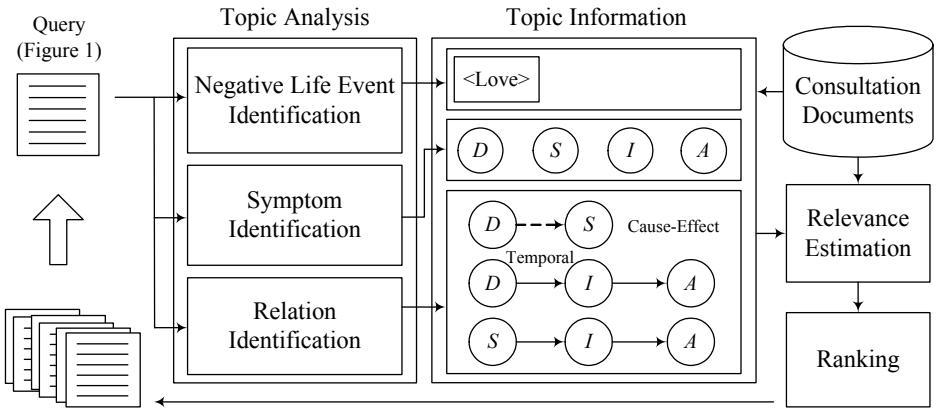


Figure 2. Framework of consultation document retrieval. The rectangle denotes a negative life event related to love relation. Each circle denotes a symptom. *D*: Depressed, *S*: Suicide, *I*: Insomnia, *A*: Anxiety.

sented by the sets of negative life events, depressive symptoms, and semantic relations. Each element in the event set and symptom set denotes an individual event and symptom, respectively, while each element in the relation set denotes a *symptom chain* to retain the order of symptoms. Similarly, the query parts of consultation documents are represented in the same manner. The relevance estimation then calculates the similarity between the input query and the query part of each consultation document by combining the similarities of the sets of events, symptoms, and relations within them. Finally, a list of consultation documents ranked in the descending order of similarities is returned to the user.

In the following, the extraction of topic information is described briefly. The detailed process is described in (Wu et al. 2005a) for symptom and relation identification, and in (Yu et al., 2007) for event identification.

1) **Symptom identification:** A total of 17 symptoms are defined based on the Hamilton Depression Rating Scale (HDRS) (Hamilton, 1960). The identification of symptoms is sentence-based. For each sentence, its structure is first analyzed by a probabilistic context free grammar (PCFG), built from the Sinica Treebank corpus developed by Academia Sinica, Taiwan (http://treebank.sinica.edu.tw), to generate a set of dependencies between word tokens. Each dependency has the format (*modifier*, *head*, $rel_{modifier,head}$). For instance, the dependency (matters, worry about, goal) means that "matters" is the goal to the head of the sentence "worry about". Each sentence can then be associated with a symptom based on the probabilities that dependencies occur in all symptoms, which are obtained from a set of training sentences.

2) **Relation Identification:** The semantic relations of interest include cause-effect and temporal relations. After the symptoms are obtained, the relations holding between symptoms (sentences) are identified by a set of discourse markers. For instance, the discourse markers "because" and "therefore" may signal cause-effect relations, and "before" and "after" may signal temporal relations.

3) **Negative life event identification:** A total of 5 types of events, namely <Family>, <Love>, <School>, <Work> and <Social> are defined based on Pagano et al's (2004) research. The identification of events is a pattern-based approach. A pattern denotes a semantically plausible combination of words, such as <parents, divorce> and <exam, fail>. First, a set of patterns is acquired from psychiatry web corpora by using an evolutionary inference algorithm. The event of each sentence is then identified by using an SVM classifier with the acquired patterns as features.

3 Retrieval Model

The similarity between a query and a document, $Sim(q,d)$, is calculated by combining the similarities of the sets of events, symptoms and relations within them, as shown in (1).

$$Sim(q,d) = \alpha Sim_{Evn}(q,d) + \beta Sim_{Sym}(q,d) + (1-\alpha-\beta) Sim_{Rel}(q,d), \quad (1)$$

where $Sim_{Evn}(q,d)$, $Sim_{Sym}(q,d)$ and $Sim_{Rel}(q,d)$, denote the similarities of the sets of events, symptoms and relations, respectively, between a query and a document, and α and β denote the combination factors.

3.1 Similarity of events and symptoms

The similarities of the sets of events and symptoms are calculated in the same method. The similarity of the event set (or symptom set) is calculated by comparing the events (or symptoms) in a query with those in a document. Additionally, only the events (or symptoms) with the same type are considered. The events (or symptoms) with different types are considered as irrelevant, i.e., no similarity. For instance, the event <Love> is considered as irrelevant to <Work>. The similarity of the event set is calculated by

$$Sim_{Evn}(q,d) = \frac{1}{N(Evn_q \cup Evn_d)} \sum_{e \in q \cap d} Type(e_q, e_d) \cos(e_q, e_d) + const., \quad (2)$$

where Evn_q and Evn_d denote the event set in a query and a document, respectively; e_q and e_d denote the events; $N(Evn_q \cup Evn_d)$ denotes the cardinality of the union of Evn_q and Evn_d as a normalization factor, and $Type(e_q, e_d)$ denotes an identity function to check whether two events have the same type, defined as

$$Type(e_q, e_d) = \begin{cases} 1 & Type(e_q) = Type(e_d) \\ 0 & \text{otherwise} \end{cases}. \quad (3)$$

The $\cos(e_q, e_d)$ denotes the cosine angle between two vectors of words representing e_q and e_d, as shown below.

$$\cos(e_q, e_d) = \frac{\sum_{i=1}^{T} w_{e_q}^i w_{e_d}^i}{\sqrt{\sum_{i=1}^{T} \left(w_{e_q}^i\right)^2} \sqrt{\sum_{i=1}^{T} \left(w_{e_d}^i\right)^2}}, \quad (4)$$

where w denotes a word in a vector, and T denotes the dimensionality of vectors. Accordingly, when two events have the same type, their similarity is given as $\cos(e_q, e_d)$ plus a constant, *const.*. Additionally, $\cos(e_q, e_d)$ and *const.* can be considered as the word-level and topic-level similarities, respectively. The optimal setting of *const.* is determined empirically.

3.2 Similarity of relations

When calculating the similarity of relations, only the relations with the same type are considered. That is, the cause-effect (or temporal) relations in a query are only compared with the cause-effect (or temporal) relations in a document. Therefore, the similarity of relation sets can be calculated as

$$Sim_{Rel}(q,d) = \frac{1}{Z} \sum_{r_q, r_d} Type(r_q, r_d) Sim(r_q, r_d), \quad (5)$$

$$Z = N_C(r_q) N_C(r_d) + N_T(r_q) N_T(r_d), \quad (6)$$

where r_q and r_d denote the relations in a query and a document, respectively; Z denotes the normalization factor for the number of relations; $Type(e_q, e_d)$ denotes an identity function similar to (3), and $N_C(\bullet)$ and $N_T(\bullet)$ denote the numbers of cause-effect and temporal relations.

Both cause-effect and temporal relations are represented by symptom chains. Hence, the similarity of relations is measured by the similarity of symptom chains. The main characteristic of a symptom chain is that it retains the cause-effect or temporal order of the symptoms within it. Therefore, the order of the symptoms must be considered when calculating the similarity of two symptom chains. Accordingly, a *sequence kernel function* (Lodhi et al., 2002; Cancedda et al., 2003) is adopted to calculate the similarity of two symptom chains. A sequence kernel compares two sequences of symbols (e.g., characters, words) based on the subsequences within them, but not individual symbols. Thereby, the order of the symptoms can be incorporated into the comparison process.

The sequence kernel calculates the similarity of two symptom chains by comparing their sub-symptom chains at different lengths. An increasing number of common sub-symptom chains indicates a greater similarity between two symptom chains. For instance, both the two symptom chains $s_1 s_2 s_3 s_4$ and $s_3 s_2 s_1$ contain the same symptoms s_1, s_2 and s_3, but in different orders. To calculate the similarity between these two symptom chains, the sequence kernel first calculates their similarities at length 2 and 3, and then averages the similarities at the two lengths. To calculate the similarity at

length 2, the sequence kernel compares their sub-symptom chains of length 2, i.e., $\{s_1s_2, s_1s_3, s_1s_4, s_2s_3, s_2s_4, s_3s_4\}$ and $\{s_3s_2, s_3s_1, s_2s_1\}$. Similarly, their similarity at length 3 is calculated by comparing their sub-symptom chains of length 3, i.e., $\{s_1s_2s_3,\ s_1s_2s_4,\ s_1s_3s_4,\ s_2s_3s_4\}$ and $\{s_3s_2s_1\}$. Obviously, no similarity exists between $s_1s_2s_3s_4$ and $s_3s_2s_1$, since no sub-symptom chains are matched at both lengths. In this example, the sub-symptom chains of length 1, i.e., individual symptoms, do not have to be compared because they contain no information about the order of symptoms. Additionally, the sub-symptom chains of length 4 do not have to be compared, because the two symptom chains share no sub-symptom chains at this length. Hence, for any two symptom chains, the length of the sub-symptom chains to be compared ranges from two to the minimum length of the two symptom chains. The similarity of two symptom chains can be formally denoted as

$$\begin{aligned} Sim(r_q, r_d) &\equiv Sim(sc_q^{N_1}, sc_d^{N_2}) \\ &= K(sc_q^{N_1}, sc_d^{N_2}) \\ &= \frac{1}{N-1}\sum_{n=2}^{N} K_n(sc_q^{N_1}, sc_d^{N_2}), \end{aligned} \tag{7}$$

where $sc_q^{N_1}$ and $sc_d^{N_2}$ denote the symptom chains corresponding to r_q and r_d, respectively; N_1 and N_2 denote the length of $sc_q^{N_1}$ and $sc_d^{N_2}$, respectively; $K(\bullet, \bullet)$ denotes the sequence kernel for calculating the similarity between two symptom chains; $K_n(\bullet, \bullet)$ denotes the sequence kernel for calculating the similarity between two symptom chains at length n, and N is the minimum length of the two symptom chains, i.e., $N = \min(N_1, N_2)$. The sequence kernel $K_n(sc_i^{N_1}, sc_j^{N_2})$ is defined as

$$\begin{aligned} K_n(sc_i^{N_1}, sc_j^{N_2}) &= \left\langle \frac{\Phi_n(sc_i^{N_1})}{\left\|\Phi_n(sc_i^{N_1})\right\|} \bullet \frac{\Phi_n(sc_j^{N_2})}{\left\|\Phi_n(sc_j^{N_2})\right\|} \right\rangle \\ &= \frac{\sum_{u \in SC^n} \phi_u(sc_i^{N_1})\phi_u(sc_j^{N_2})}{\sqrt{\sum_{u \in SC^n} \phi_u(sc_i^{N_1})\phi_u(sc_j^{N_1})}\sqrt{\sum_{u \in SC^n} \phi_u(sc_i^{N_2})\phi_u(sc_j^{N_2})}}, \end{aligned} \tag{8}$$

where $K_n(sc_i^{N_1}, sc_j^{N_2})$ is the normalized inner product of vectors $\Phi_n(sc_i^{N_1})$ and $\Phi_n(sc_j^{N_2})$; $\Phi_n(\bullet)$ denotes a mapping that transforms a given symptom chain into a vector of the sub-symptom chains of length n; $\phi_u(\bullet)$ denotes an element of the vector, representing the weight of a sub-symptom chain u, and SC^n denotes the set of all possible sub-symptom chains of length n. The weight of a sub-symptom chain, i.e., $\phi_u(\bullet)$, is defined as

$$\phi_u(sc_i^{N_1}) = \begin{cases} 1 & u \text{ is a contiguous sub-symptom chain of } sc_i^{N_1} \\ \lambda^{\theta} & u \text{ is a non-contiguous sub-symptom chain with } \theta \text{ skipped symptoms} \\ 0 & u \text{ does not appear in } sc_i^{N_1}, \end{cases} \tag{9}$$

where $\lambda \in [0,1]$ denotes a *decay factor* that is adopted to penalize the non-contiguous sub-symptom chains occurred in a symptom chain based on the skipped symptoms. For instance, $\phi_{s_1s_2}(s_1s_2s_3) = \phi_{s_2s_3}(s_1s_2s_3) = 1$ since s_1s_2 and s_2s_3 are considered as contiguous in $s_1s_2s_3$, and $\phi_{s_1s_3}(s_1s_2s_3) = \lambda^1$ since s_1s_3 is a non-contiguous sub-symptom chain with one skipped symptom. The decay factor is adopted because a contiguous sub-symptom chain is preferable to a non-contiguous chain when comparing two symptom chains. The setting of the decay factor is domain dependent. If $\lambda = 1$, then no penalty is applied for skipping symptoms, and the cause-effect and temporal relations are transitive. The optimal setting of

Given the two symptom chains, $s_1s_2s_3$ and $s_1s_2s_3s_4$.
Their similarity is calculated as

$$K(s_1s_2s_3, s_1s_2s_3s_4) = \frac{1}{2}\sum_{n=2}^{3} K_n(s_1s_2s_3, s_1s_2s_3s_4).$$

$$\begin{aligned} K_2(s_1s_2s_3, s_1s_2s_3s_4) &= \left\langle \frac{\Phi_2(s_1s_2s_3)}{\left\|\Phi_2(s_1s_2s_3)\right\|} \bullet \frac{\Phi_2(s_1s_2s_3s_4)}{\left\|\Phi_2(s_1s_2s_3s_4)\right\|} \right\rangle \\ &= \frac{2+\lambda^2}{\sqrt{(3+2\lambda^2+\lambda^4)(2+\lambda^2)}} = 0.71(\lambda = 1) \end{aligned}$$

Similarly, $K_3(s_1s_2s_3, s_1s_2s_3s_4) = 0.5$.

	$\phi_{s_1s_2}$	$\phi_{s_1s_3}$	$\phi_{s_1s_4}$	$\phi_{s_2s_3}$	$\phi_{s_2s_4}$	$\phi_{s_3s_4}$	...	$\phi_{s_{16}s_{17}}$	
$\Phi_2(s_1s_2s_3) = ($	1	λ	0	1	0	0	...	0	$)$
$\Phi_2(s_1s_2s_3s_4) = ($	1	λ	λ^2	1	λ	1	...	0	$)$

$\Phi_2(s_1s_2s_3) \bullet \Phi_2(s_1s_2s_3s_4) = 2+\lambda^2$
$\Phi_2(s_1s_2s_3) \bullet \Phi_2(s_1s_2s_3) - 2+\lambda^2$
$\Phi_2(s_1s_2s_3s_4) \bullet \Phi_2(s_1s_2s_3s_4) = 3+2\lambda^2+\lambda^4$

Figure 3. Illustrative example of relevance computation using the sequence kernel function.

Topic	Avg. Number
Negative Life Event	1.45
Depressive Symptom	4.40
Semantic Relation	3.35

Table 1. Characteristics of the test query set.

λ is determined empirically. Figure 3 presents an example to summarize the computation of the similarity between two symptom chains.

4 Experimental Results

4.1 Experiment setup

1) **Corpus:** The consultation documents were collected from the mental health website of the John Tung Foundation (http://www.jtf.org.tw) and the PsychPark (http://www.psychpark.org), a virtual psychiatric clinic, maintained by a group of volunteer professionals of Taiwan Association of Mental Health Informatics (Bai et al. 2001). Both of the web sites provide various kinds of free psychiatric services and update the consultation documents periodically. For privacy consideration, all personal information has been removed. A total of 3,650 consultation documents were collected for evaluating the retrieval model, of which 20 documents were randomly selected as the test query set, 100 documents were randomly selected as the tuning set to obtain the optimal parameter settings of involved retrieval models, and the remaining 3,530 documents were the reference set to be retrieved. Table 1 shows the average number of events, symptoms and relations in the test query set.

2) **Baselines:** The proposed method, denoted as Topic, was compared to two word-based retrieval models: the VSM and Okapi BM25 models. The VSM was implemented in terms of the standard TF-IDF weight. The Okapi BM25 model is defined as

$$\sum_{t \in Q} w^{(1)} \frac{(k_1+1)tf}{K+tf} \frac{(k_3+1)qtf}{k_3+qtf} + k_2 |Q| \frac{avdl-dl}{avdl+dl}, \quad (10)$$

where t denotes a word in a query Q; qtf and tf denote the word frequencies occurring in a query and a document, respectively, and $w^{(1)}$ denotes the Robertson-Sparck Jones weight of t (without relevance feedback), defined as

$$w^{(1)} = \log \frac{N-n+0.5}{n+0.5}, \quad (11)$$

where N denotes the total number of documents, and n denotes the number of documents containing t. In (10), K is defined as

$$K = k_1((1-b)+b \cdot dl/avdl), \quad (12)$$

where dl and $avdl$ denote the length and average length of a document, respectively. The default values of k_1, k_2, k_3 and b are describe in (Robertson et al., 1996), where k_1 ranges from 1.0 to 2.0; k_2 is set to 0; k_3 is set to 8, and b ranges from 0.6 to 0.75. Additionally, BM25 can be considered as BM15 and BM11 when b is set to 1 and 0, respectively.

3) **Evaluation metric:** To evaluate the retrieval models, a multi-level relevance criterion was adopted. The relevance criterion was divided into four levels, as described below.
   - Level 0: No topics are matched between a query and a document.
   - Level 1: At least one topic is partially matched between a query and a document.
   - Level 2: All of the three topics are partially matched between a query and a document.
   - Level 3: All of the three topics are partially matched, and at least one topic is exactly matched between a query and a document.

   To deal with the multi-level relevance, the *discounted cumulative gain (DCG)* (Jarvelin and Kekalainen, 2000) was adopted as the evaluation metric, defined as

$$DCG[i] = \begin{cases} G[1], & \text{if } i = 1 \\ DCG[i-1] + G[i]/\log_c i, & \text{otherwise} \end{cases} \quad (13)$$

where i denotes the i-th document in the retrieved list; $G[i]$ denotes the gain value, i.e., relevance levels, of the i-th document, and c denotes the parameter to penalize a retrieved document in a lower rank. That is, the DCG simultaneously considers the relevance levels, and the ranks in the retrieved list to measure the retrieval precision. For instance, let <3,2,3,0,0> denotes the retrieved list of five documents with their relevance levels. If no penalization is used, then the DCG values for

Relevance Level	Avg. Number
Level 1	18.50
Level 2	9.15
Level 3	2.20

Table 2. Average number of relevant documents for the test query set.

Retrieval Model	Avg. Time (seconds)
Topic	17.13
VSM	0.68
BM25	0.48

Table 4. Average query processing time of different retrieval models.

	DCG(5)	DCG(10)	DCG(20)	DCG(50)	DCG(100)
Topic	4.7516*	6.9298	7.6040*	8.3606*	9.3974*
BM25	4.4624	6.7023	7.1156	7.8129	8.6597
BM11	3.8877	4.9328	5.9589	6.9703	7.7057
VSM	2.3454	3.3195	4.4609	5.8179	6.6945
BM15	2.1362	2.6120	3.4487	4.5452	5.7020

Table 3. DCG values of different retrieval models. * Topic vs BM25 significantly different (p<0.05)

the retrieved list are <3,5,8,8,8>, and thus DCG[5]=8. Conversely, if c=2, then the documents retrieved at ranks lower than two are penalized. Hence, the DCG values for the retrieved list are <3,5,6.89,6.89,6.89>, and DCG[5]=6.89.

The relevance judgment was performed by three experienced physicians. First, the *pooling* method (Voorhees, 2000) was adopted to generate the candidate relevant documents for each test query by taking the top 50 ranked documents retrieved by each of the involved retrieval models, namely the VSM, BM25 and Topic. Two physicians then judged each candidate document based on the multilevel relevance criterion. Finally, the documents with disagreements between the two physicians were judged by the third physician. Table 2 shows the average number of relevant documents for the test query set.

4) **Optimal parameter setting:** The parameter settings of BM25 and Topic were evaluated using the tuning set. The optimal setting of BM25 were k_1 =1 and b=0.6. The other two parameters were set to the default values, i.e., $k_2 = 0$ and $k_3 = 8$. For the Topic model, the parameters required to be evaluated include the combination factors, α and β, described in (1); the constant *const.* described in (2), and the decay factor, λ, described in (9). The optimal settings were $\alpha = 0.3$; $\beta = 0.5$; *const.*=0.6 and $\lambda = 0.8$.

4.2 Retrieval results

The results are divided into two groups: the *precision* and *efficiency*. The retrieval precision was measured by DCG values. Additionally, a paired, two-tailed *t-test* was used to determine whether the performance difference was statistically significant. The retrieval efficiency was measure by the query processing time, i.e., the time for processing all the queries in the test query set.

Table 3 shows the comparative results of retrieval precision. The two variants of BM25, namely BM11 and BM15, are also considered in comparison. For the word-based retrieval models, both BM25 and BM11 outperformed the VSM, and BM15 performed worst. The Topic model achieved higher DCG values than both the BM-series models and VSM. The reasons are three-fold. First, a negative life event and a symptom can each be expressed by different words with the same or similar meaning. Therefore, the word-based models often failed to retrieve the relevant documents when different words were used in the input query. Second, a word may relate to different events and symptoms. For instance, the term "worry about" is

a good indicator for both the symptoms <Anxiety> and <Hypochondriasis>. This may result in ambiguity for the word-based models. Third, the word-based models cannot capture semantic relations between symptoms. The Topic model incorporates not only the word-level information, but also more useful topic information about depressive problems, thus improving the retrieval results.

The query processing time was measured using a personal computer with Windows XP operating system, a 2.4GHz Pentium IV processor and 512MB RAM. Table 4 shows the results. The topic model required more processing time than both VSM and BM25, since identification of topics involves more detailed analysis, such as semantic parsing of sentences and symptom chain construction. This finding indicates that although the topic information can improve the retrieval precision, incorporating such high-precision features reduces the retrieval efficiency.

5 Conclusion

This work has presented the use of topic information for retrieving psychiatric consultation documents. The topic information can provide more precise information about users' depressive problems, thus improving the retrieval precision. The proposed framework can also be applied to different domains as long as the domain-specific topic information is identified. Future work will focus on more detailed experiments, including the contribution of each topic to retrieval precision, the effect of using different methods to combine topic information, and the evaluation on real users.

References

Baeza-Yates, R. and B. Ribeiro-Neto. 1999. *Modern Information Retrieval*. Addison-Wesley, Reading, MA.

Cancedda, N., E. Gaussier, C. Goutte, and J. M. Renders. 2003. Word-Sequence Kernels. *Journal of Machine Learning Research*, 3(6):1059-1082.

Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database.* MIT Press, Cambridge, MA.

Hamilton, M. 1960. A Rating Scale for Depression. *Journal of Neurology, Neurosurgery and Psychiatry*, 23:56-62

Jarvelin, K. and J. Kekalainen. 2000. IR Evaluation Methods for Retrieving Highly Relevant Documents. *In Proc. of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 41-48.

Lodhi, H., C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. 2002. Text Classification Using String Kernels. *Journal of Machine Learning Research*, 2(3):419-444.

Okabe, M., K. Umemura and S. Yamada. 2005. Query Expansion with the Minimum User Feedback by Transductive Learning. In *Proc. of HLT/EMNLP*, Vancouver, Canada, pages 963-970.

Pagano, M.E., A.E. Skodol, R.L. Stout, M.T. Shea, S. Yen, C.M. Grilo, C.A. Sanislow, D.S. Bender, T.H. McGlashan, M.C. Zanarini, and J.G. Gunderson. 2004. Stressful Life Events as Predictors of Functioning: Findings from the Collaborative Longitudinal Personality Disorders Study. *Acta Psychiatrica Scandinavica*, 110: 421-429.

Robertson, S. E., S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M.Gatford. 1995. Okapi at TREC-3. In *Proc. of the Third Text REtrieval Conference (TREC-3)*, NIST.

Robertson, S. E., S. Walker, M. M. Beaulieu, and M.Gatford. 1996. Okapi at TREC-4. In *Proc. of the fourth Text REtrieval Conference (TREC-4)*, NIST.

Voorhees, E. M. and D. K. Harman. 2000. Overview of the Sixth Text REtrieval Conference (TREC-6). *Information Processing and Management*, 36(1):3-35.

Wu, C. H., L. C. Yu, and F. L. Jang. 2005a. Using Semantic Dependencies to Mine Depressive Symptoms from Consultation Records. *IEEE Intelligent System*, 20(6):50-58.

Wu, C. H., J. F. Yeh, and M. J. Chen. 2005b. Domain-Specific FAQ Retrieval Using Independent Aspects. *ACM Trans. Asian Language Information Processing*, 4(1):1-17.

Wu, C. H., J. F. Yeh, and Y. S. Lai. 2006. Semantic Segment Extraction and Matching for Internet FAQ Retrieval. *IEEE Trans. Knowledge and Data Engineering*, 18(7):930-940.

Yeh, J. F., C. H. Wu, M. J. Chen, and L. C. Yu. 2004. Automated Alignment and Extraction of Bilingual Domain Ontology for Cross-Language Domain-Specific Applications. In *Proc. of the 20th COLING*, Geneva, Switzerland, pages 1140-1146.

Yu, L. C., C. H. Wu, Yeh, J. F., and F. L. Jang. 2007. HAL-based Evolutionary Inference for Pattern Induction from Psychiatry Web Resources. Accepted by *IEEE Trans. Evolutionary Computation.*

What to be? - Electronic Career Guidance Based on Semantic Relatedness

Iryna Gurevych, Christof Müller and Torsten Zesch
Ubiquitous Knowledge Processing Group
Telecooperation, Darmstadt University of Technology
Hochschulstr. 10, 64289 Darmstadt, Germany
http://www.ukp.tu-darmstadt.de
{gurevych,mueller,zesch}@tk.informatik.tu-darmstadt.de

Abstract

We present a study aimed at investigating the use of semantic information in a novel NLP application, Electronic Career Guidance (ECG), in German. ECG is formulated as an information retrieval (IR) task, whereby textual descriptions of professions (*documents*) are ranked for their relevance to natural language descriptions of a person's professional interests (*the topic*). We compare the performance of two semantic IR models: (IR-1) utilizing semantic relatedness (SR) measures based on either wordnet or Wikipedia and a set of heuristics, and (IR-2) measuring the similarity between the topic and documents based on Explicit Semantic Analysis (ESA) (Gabrilovich and Markovitch, 2007). We evaluate the performance of SR measures intrinsically on the tasks of (T-1) computing SR, and (T-2) solving Reader's Digest Word Power (RDWP) questions.

1 Electronic Career Guidance

Career guidance is important both for the person involved and for the state. Not well informed decisions may cause people to drop the training program they are enrolled in, yielding loss of time and financial investments. However, there is a mismatch between what people know about existing professions and the variety of professions, which exist in reality. Some studies report that school leavers typically choose the professions known to them, such as *policeman*, *nurse*, etc. Many other professions, which can possibly match the interests of the person very well, are not chosen, as their titles are unknown and people seeking career advice do not know about their existence, e.g. *electronics installer*, or *chemical laboratory worker*. However, people are very good at describing their professional interests in natural language. That is why they are even asked to write a short essay prior to an appointment with a career guidance expert.

Electronic career guidance is, thus, a supplement to career guidance by human experts, helping young people to decide which profession to choose. The goal is to automatically compute a ranked list of professions according to the user's interests. A current system employed by the German Federal Labour Office (GFLO) in their automatic career guidance front-end[1] is based on vocational trainings, manually annotated using a tagset of 41 keywords. The user must select appropriate keywords according to her interests. In reply, the system consults a knowledge base with professions manually annotated with the keywords by career guidance experts. Thereafter, it outputs a list of the best matching professions to the user. This approach has two significant disadvantages. Firstly, the knowledge base has to be maintained and steadily updated, as the number of professions and keywords associated with them is continuously changing. Secondly, the user has to describe her interests in a very restricted way.

At the same time, GFLO maintains an extensive database with textual descriptions of professions,

[1] http://www.interesse-beruf.de/

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 1032–1039,
Prague, Czech Republic, June 2007.

called BERUFEnet.[2] Therefore, we cast the problem of ECG as an IR task, trying to remove the disadvantages of conventional ECG outlined above by letting the user describe her interests in a short natural language essay, called *a professional profile*.

Example essay translated to English
I would like to work with animals, to treat and look after them, but I cannot stand the sight of blood and take too much pity on them. On the other hand, I like to work on the computer, can program in C, Python and VB and so I could consider software development as an appropriate profession. I cannot imagine working in a kindergarden, as a social worker or as a teacher, as I am not very good at asserting myself.

Textual descriptions of professions are ranked given such an essay by using NLP and IR techniques. As essays and descriptions of professions display a mismatch between the vocabularies of topics and documents and there is lack of contextual information, due to the documents being fairly short as compared to standard IR scenarios, lexical semantic information should be especially beneficial to an IR system. For example, the profile can contain words about some objects or activities related to the profession, but not directly mentioned in the description, e.g. *oven*, *cakes* in the profile and *pastries*, *baker*, or *confectioner* in the document. Therefore, we propose to utilize semantic relatedness as a ranking function instead of conventional IR techniques, as will be substantiated below.

2 System Architecture

Integrating lexical semantic knowledge in ECG requires the existence of knowledge bases encoding domain and lexical knowledge. In this paper, we investigate the utility of two knowledge bases: (i) a German wordnet, GermaNet (Kunze, 2004), and (ii) the German portion of Wikipedia.[3] A large body of research exists on using wordnets in NLP applications and in particular in IR (Moldovan and Mihalcea, 2000). The knowledge in wordnets has been typically utilized by expanding queries with related terms (Vorhees, 1994; Smeaton et al., 1994), concept indexing (Gonzalo et al., 1998), or similarity measures as ranking functions (Smeaton et al., 1994; Müller and Gurevych, 2006). Recently, Wikipedia has been discovered as a promising lexical semantic resource and successfully used in such different NLP tasks as question answering (Ahn et al., 2004), named entity disambiguation (Bunescu and Pasca, 2006), and information retrieval (Katz et al., 2005). Further research (Zesch et al., 2007b) indicates that German wordnet and Wikipedia show different performance depending on the task at hand.

Departing from this, we first compare two semantic relatedness (SR) measures based on the information either in the German wordnet (Lin, 1998) called **LIN**, or in Wikipedia (Gabrilovich and Markovitch, 2007) called Explicit Semantic Analysis, or **ESA**. We evaluate their performance intrinsically on the tasks of (T-1) computing semantic relatedness, and (T-2) solving Reader's Digest Word Power (RDWP) questions and make conclusions about the ability of the measures to model certain aspects of semantic relatedness and their coverage. Furthermore, we follow the approach by Müller and Gurevych (2006), who proposed to utilize the LIN measure and a set of heuristics as an IR model (IR-1).

Additionally, we utilize the ESA measure in a semantic information retrieval model, as this measure is significantly better at vocabulary coverage and at modelling cross part-of-speech relations (Gabrilovich and Markovitch, 2007). We compare the performance of ESA and LIN measures in a task-based IR evaluation and analyze their strengths and limitations. Finally, we apply ESA to directly compute text similarities between topics and documents (IR-2) and compare the performance of two semantic IR models and a baseline Extended Boolean (EB) model (Salton et al., 1983) with query expansion.[4]

To summarize, the *contributions of this paper* are three-fold: (i) we present a novel system, utilizing NLP and IR techniques to perform Electronic Career Guidance, (ii) we study the properties and intrinsically evaluate two SR measures based on GermaNet and Wikipedia for the tasks of computing semantic relatedness and solving Reader's Digest Word Power Game questions, and (iii) we investigate the performance of two semantic IR models in a task based evaluation.

[2] http://infobub.arbeitsagentur.de/berufe/

[3] http://de.wikipedia.org/

[4] We also ran experiments with Okapi BM25 model as implemented in the Terrier framework, but the results were worse than those with the EB model. Therefore, we limit our discussion to the latter.

3 Computing Semantic Relatedness

3.1 SR Measures

GermaNet based measures GermaNet is a German wordnet, which adopted the major properties and database technology from Princeton's WordNet (Fellbaum, 1998). However, GermaNet displays some structural differences and content oriented modifications. Its designers relied mainly on linguistic evidence, such as corpus frequency, rather than psycholinguistic motivations. Also, GermaNet employs artificial, i.e. non-lexicalized concepts, and adjectives are structured hierarchically as opposed to WordNet. Currently, GermaNet includes about 40000 synsets with more than 60000 word senses modelling nouns, verbs and adjectives.

We use the semantic relatedness measure by Lin (1998) (referred to as LIN), as it consistently is among the best performing wordnet based measures (Gurevych and Niederlich, 2005; Budanitsky and Hirst, 2006). Lin defined semantic similarity using a formula derived from information theory. This measure is sometimes called a universal semantic similarity measure as it is supposed to be application, domain, and resource independent. *Lin* is computed as:

$$sim_{c_1,c_2} = \frac{2 \times \log p(LCS(c_1, c_2))}{\log p(c_1) + \log p(c_2)}$$

where c_1 and c_2 are concepts (word senses) corresponding to w_1 and w_2, $\log p(c)$ is the information content, and $LCS(c_1, c_2)$ is the lowest common subsumer of the two concepts. The probability p is computed as the relative frequency of words (representing that concept) in the taz[5] corpus.

Wikipedia based measures Wikipedia is a free online encyclopedia that is constructed in a collaborative effort of voluntary contributors and still grows exponentially. During this process, Wikipedia has probably become the largest collection of freely available knowledge. Wikipedia shares many of its properties with other well known lexical semantic resources (like dictionaries, thesauri, semantic wordnets or conventional encyclopedias) (Zesch et al., 2007a). As Wikipedia also models relatedness between concepts, it is better suited for computing semantic relatedness than GermaNet (Zesch et al., 2007b).

In very recent work, Gabrilovich and Markovitch (2007) introduce a SR measure called *Explicit Semantic Analysis* (ESA). The ESA measure represents the meaning of a term as a high-dimensional concept vector. The concept vector is derived from Wikipedia articles, as each article focuses on a certain topic, and can thus be viewed as expressing a concept. The dimension of the concept vector is the number of Wikipedia articles. Each element of the vector is associated with a certain Wikipedia article (or concept). If the term can be found in this article, the term's tfidf score (Salton and McGill, 1983) in this article is assigned to the vector element. Otherwise, 0 is assigned. As a result, a term's concept vector represents the importance of the term for each concept. Semantic relatedness of two terms can then be easily computed as the cosine of their corresponding concept vectors. If we want to measure the semantic relatedness of texts instead of terms, we can also use ESA concept vectors. A text is represented as the average concept vector of its terms' concept vectors. Then, the relatedness of two texts is computed as the cosine of their average concept vectors.

As ESA uses all textual information in Wikipedia, the measure shows excellent coverage. Therefore, we select it as the second measure for integration into our IR system.

3.2 Datasets

Semantic relatedness datasets for German employed in our study are presented in Table 1. Gurevych (2005) conducted experiments with two datasets: i) a German translation of the English dataset by Rubenstein and Goodenough (1965) (**Gur65**), and ii) a larger dataset containing 350 word pairs (**Gur350**). Zesch and Gurevych (2006) created a third dataset from domain-specific corpora using a semi-automatic process (**ZG222**). Gur65 is rather small and contains only noun-noun pairs connected by either synonymy or hypernymy. Gur350 contains nouns, verbs and adjectives that are connected by classical and non-classical relations (Morris and Hirst, 2004). However, word pairs for this dataset are biased towards strong classical relations, as they were manually selected from a corpus.

[5] http://www.taz.de

Dataset	Year	Language	# Pairs	POS	Scores	# Subjects	Correlation r Inter	Intra
Gur65	2005	German	65	N	discrete {0,1,2,3,4}	24	.810	-
Gur350	2006	German	350	N, V, A	discrete {0,1,2,3,4}	8	.690	-
ZG222	2006	German	222	N, V, A	discrete {0,1,2,3,4}	21	.490	.647

Table 1: Comparison of datasets used for evaluating semantic relatedness in German.

ZG222 does not have this bias.

Following the work by Jarmasz and Szpakowicz (2003) and Turney (2006), we created a second dataset containing multiple choice questions. We collected 1072 multiple-choice word analogy questions from the German **Reader's Digest Word Power Game** (RDWP) from January 2001 to December 2005 (Wallace and Wallace, 2005). We discarded 44 questions that had more than one correct answer, and 20 questions that used a phrase instead of a single term as query. The resulting 1008 questions form our evaluation dataset. An example question is given below:

Muffin (**muffin**)
a) Kleingebäck (small cake)
b) Spenglerwerkzeug (plumbing tool)
c) Miesepeter (killjoy)
d) Wildschaf (moufflon)

The task is to find the correct choice - 'a)' in this case.

This dataset is significantly larger than any of the previous datasets employed in this type of evaluation. Also, it is not restricted to synonym questions, as in the work by Jarmasz and Szpakowicz (2003), but also includes hypernymy/hyponymy, and few non-classical relations.

3.3 Analysis of Results

Table 2 gives the results of evaluation on the task of correlating the results of an SR measure with human judgments using Pearson correlation. The GermaNet based LIN measure outperforms ESA on the Gur65 dataset. On the other datasets, ESA is better than LIN. This is clearly due to the fact, that Gur65 contains only noun-noun word pairs connected by classical semantic relations, while the other datasets also contain cross part-of-speech pairs connected by non-classical relations. The Wikipedia based ESA measure can better capture such relations. Additionally, Table 3 shows that ESA also covers almost all word pairs in each dataset, while GermaNet is much lower for Gur350 and ZG222. ESA performs even better on the Reader's Digest task (see Table 4). It shows high coverage and near human performance regarding the relative number of correctly solved questions.[6] Given the high performance and coverage of the Wikipedia based ESA measure, we expect it to yield better IR results than LIN.

	Gur65	Gur350	ZG222
# covered word pairs	53	116	55
Upper bound	0.80	0.64	0.44
GermaNet Lin	0.73	0.50	0.08
Wikipedia ESA	0.56	0.52	0.32

Table 2: Pearson correlation r of human judgments with SR measures on word pairs covered by GermaNet and Wikipedia.

Dataset	# Pairs	Covered Pairs Lin	ESA
Gur65	65	60	65
Gur350	350	208	333
ZG222	222	88	205

Table 3: Number of covered word pairs based on Lin or ESA measure on different datasets.

4 Information Retrieval

4.1 IR Models

Preprocessing For creating the search index for IR models, we apply first tokenization and then remove stop words. We use a general German stop

[6] Values for human performance are for one subject. Thus, they only indicate the approximate difficulty of the task. We plan to use this dataset with a much larger group of subjects.

	#Answered	#Correct	Ratio
Human	1008	874	0.87
GermaNet Lin	298	153	0.51
Wikipedia ESA	789	572	0.72

Table 4: Evaluation results on multiple-choice word analogy questions.

word list extended with highly frequent domain specific terms. Before adding the remaining words to the index, they are lemmatized. We finally split compounds into their constituents, and add both, constituents and compounds, to the index.

EB model Lucene[7] is an open source text search library based on an EB model. After matching the preprocessed queries against the index, the document collection is divided into a set of relevant and irrelevant documents. The set of relevant documents is, then, ranked according to the formula given in the following equation:

$$r_{EB}(d,q) = \sum_{i=1}^{n_q} tf(t_q,d) \cdot idf(t_q) \cdot lengthNorm(d)$$

where n_q is the number of terms in the query, $tf(t_q,d)$ is the term frequency factor for term t_q in document d, $idf(t_q)$ is the inverse document frequency of the term, and $lengthNorm(d)$ is a normalization value of document d, given the number of terms within the document. We added a simple query expansion algorithm using (i) synonyms, and (ii) hyponyms, extracted from GermaNet.

IR based on SR For the (IR-1) model, we utilize two SR measures and a set of heuristics: (i) the Lin measure based on GermaNet (LIN), and (ii) the ESA measure based on Wikipedia (ESA-Word). This algorithm was applied to the German IR benchmark with positive results by Müller and Gurevych (2006). The algorithm computes a SR score for each query and document term pair. Scores above a predefined threshold are summed up and weighted by different factors, which boost or lower the scores for documents, depending on how many query terms are contained exactly or contribute a high enough SR score. In order to integrate the strengths of traditional IR models, the inverse document frequency *idf* is considered, which measures the general importance of a term for predicting the content of a document. The final formula of the model is as follows:

$$r_{SR}(d,q) = \frac{\sum_{i=1}^{n_d} \sum_{j=1}^{n_q} idf(t_{q,j}) \cdot s(t_{d,i}, t_{q,j})}{(1+n_{nsm}) \cdot (1+n_{nr})}$$

where n_d is the number of tokens in the document, n_q the number of tokens in the query, $t_{d,i}$ the i-th document token, $t_{q,j}$ the j-th query token, $s(t_{d,i}, t_{q,j})$ the SR score for the respective document and query term, n_{nsm} the number of query terms not exactly contained in the document, n_{nr} the number of query tokens, which do not contribute a SR score above the threshold.

For the (IR-2) model, we apply the ESA method for directly comparing the query with documents, as described in Section 3.1.

4.2 Data

The corpus employed in our experiments was built based on a real-life IR scenario in the domain of ECG, as described in Section 1. The **document collection** is extracted from BERUFEnet,[8] a database created by the GFLO. It contains textual descriptions of about 1,800 vocational trainings, and 4,000 descriptions of professions. We restrict the collection to a subset of BERUFEnet documents, consisting of 529 descriptions of vocational trainings, due to the process necessary to obtain relevance judgments, as described below. The documents contain not only details of professions, but also a lot of information concerning the training and administrative issues. We only use those portions of the descriptions, which characterize the profession itself.

We collected real natural language **topics** by asking 30 human subjects to write an essay about their professional interests. The topics contain 130 words, on average. Making **relevance judgments** for ECG requires domain expertise. Therefore, we applied an automatic method, which uses the knowledge base employed by the GFLO, described in Section 1. To obtain relevance judgments, we first annotate each essay with relevant keywords from the tagset of 41 and retrieve a ranked list of professions, which were assigned one or more keywords by domain experts. To map the ranked list to a set of relevant and irrelevant professions, we use a threshold of 3, as suggested by career guidance experts. This setting yields on average 93 relevant documents per topic. The quality of the automatically created gold standard depends on the quality of the applied knowledge base. As the knowledge base was created by

[7] http://lucene.apache.org

[8] http://berufenet.arbeitsamt.de/

domain experts and is at the core of the electronic career guidance system of the GFLO, we assume that the quality is adequate to ensure a reliable evaluation.

4.3 Analysis of Results

In Table 5, we summarize the results of the experiments applying different IR models on the BERUFEnet data. We build queries from natural language essays by (QT-1) extracting nouns, verbs, and adjectives, (QT-2) using only nouns, and (QT-3) manually assigning suitable keywords from the tagset with 41 keywords to each topic. We report the results with two different thresholds (.85 and .98) for the Lin model, and with three different thresholds (.11, .13 and .24) for the ESA-Word models. The evaluation metrics used are *mean average precision* (MAP), *precision after ten documents* (P10), *the number of relevant returned documents* (#RRD). We compute the absolute value of *Spearman's rank correlation coefficient* (SRCC) by comparing the relevance ranking of our system with the relevance ranking of the knowledge base employed by the GFLO.

Using query expansion for the EB model decreases the retrieval performance for most configurations. The SR based models outperform the EB model in all configurations and evaluation metrics, except for P10 on the keyword based queries. The Lin model is always outperformed by at least one of the ESA models, except for (QT-3). (IR-2) performs best on longer queries using nouns, verbs, adjectives or just nouns.

Comparing the number of relevant retrieved documents, we observe that the IR models based on SR are able to return more relevant documents than the EB model. This supports the claim that semantic knowledge is especially helpful for the vocabulary mismatch problem, which cannot be addressed by conventional IR models. E.g., only SR-based models can find the job *information technician* for a profile which contains the sentence *My interests and skills are in the field of languages and IT*. The job could only be judged as relevant, as the semantic relation between *IT* in the profile and *information technology* in the professional description could be found.

In our analysis of the BERUFEnet results obtained on (QT-1), we noticed that many errors were due to the topics expressed in free natural language essays. Some subjects deviated from the given task to describe their professional interests and described facts that are rather irrelevant to the task of ECG, e.g. *It is important to speak different languages in the growing European Union*. If all content words are extracted to build a query, a lot of noise is introduced.

Therefore, we conducted further experiments with (QT-2) and (QT-3): building the query using only nouns, and using manually assigned keywords based on the tagset of 41 keywords. For example, the following query is built for the professional profile given in Section 1.

```
Keywords assigned:
care for/nurse/educate/teach; use/program computer;
office; outside:  outside facilities/natural
environment; animals/plants
```

IR results obtained on (QT-2) and (QT-3) show that the performance is better for nouns, and significantly better for the queries built of keywords. This suggests that in order to achieve high IR performance for the task of Electronic Career Guidance, it is necessary to preprocess the topics by performing information extraction to remove the noise from free text essays. As a result of the preprocessing, natural language essays should be mapped to a set of keywords relevant for describing a person's interests. Our results suggest that the word-based semantic relatedness IR model (IR-1) performs significantly better in this setting.

5 Conclusions

We presented a system for Electronic Career Guidance utilizing NLP and IR techniques. Given a natural language professional profile, relevant professions are computed based on the information about semantic relatedness. We intrinsically evaluated and analyzed the properties of two semantic relatedness measures utilizing the lexical semantic information in a German wordnet and Wikipedia on the tasks of estimating semantic relatedness scores and answering multiple-choice questions. Furthermore, we applied these measures to an IR task, whereby they were used either in combination with a set of heuristics or the Wikipedia based measure was used to directly compute semantic relatedness of topics and

Model	(QT-1) Nouns, Verbs, Adj.				(QT-2) Nouns				(QT-3) Keywords			
	MAP	P10	#RRD	SRCC	MAP	P10	#RRD	SRCC	MAP	P10	#RRD	SRCC
EB	.39	.58	2581	.306	.38	.58	2297	.335	.54	.76	2755	.497
EB+SYN	.37	.56	2589	.288	.38	.57	2310	.331	.54	.73	2768	.530
EB+HYPO	.34	.47	2702	.275	.38	.56	2328	.327	.47	.65	2782	.399
Lin .85	.41	.56	**2787**	.338	.40	.59	2770	.320	.59	.73	**2787**	**.578**
Lin .98	.41	.61	2753	.326	.42	.59	2677	.341	.58	.74	2783	.563
ESA-Word .11	.39	.56	**2787**	.309	.44	.63	**2787**	.355	.60	**.77**	**2787**	.535
ESA-Word .13	.38	.59	**2787**	.282	.43	.62	**2787**	.338	**.62**	.76	**2787**	**.550**
ESA-Word .24	.40	.60	**2787**	.259	.43	.60	2699	.306	.54	.73	2772	.482
ESA-Text	**.47**	**.62**	**2787**	**.368**	**.55**	**.71**	**2787**	**.462**	.56	.74	**2787**	.489

Table 5: Information Retrieval performance on the BERUFEnet dataset.

documents. We experimented with three different query types, which were built from the topics by: (QT-1) extracting nouns, verbs, adjectives, (QT-2) extracting only nouns, or (QT-3) manually assigning several keywords to each topic from a tagset of 41 keywords.

In an intrinsic evaluation of LIN and ESA measures on the task of computing semantic relatedness, we found that ESA captures the information about semantic relatedness and non-classical semantic relations considerably better than LIN, which operates on an *is-a* hierarchy and, thus, better captures the information about semantic similarity. On the task of solving RDWP questions, the ESA measure significantly outperformed the LIN measure in terms of correctness. On both tasks, the coverage of ESA is much better. Despite this, the performance of LIN and ESA as part of an IR model is only slightly different. ESA performs better for all lengths of queries, but the differences are not as significant as in the intrinsic evaluation. This indicates that the information provided by both measures, based on different knowledge bases, might be complementary for the IR task.

When ESA is applied to directly compute semantic relatedness between topics and documents, it outperforms IR-1 and the baseline EB model by a large margin for QT-1 and QT-2 queries. For QT-3, i.e., the shortest type of query, it performs worse than IR-1 utilizing ESA and a set of heuristics. Also, the performance of the baseline EB model is very strong in this experimental setting. This result indicates that IR-2 utilizing conventional information retrieval techniques and semantic information from Wikipedia is better suited for longer queries providing enough context. For shorter queries, *soft* matching techniques utilizing semantic relatedness tend to be beneficial.

It should be born in mind, that the construction of QT-3 queries involved a manual step of assigning the keywords to a given essay. In this experimental setting, all models show the best performance. This indicates that professional profiles contain a lot of noise, so that more sophisticated NLP analysis of topics is required. This will be improved in our future work, whereby the system will incorporate an information extraction component for automatically mapping the professional profile to a set of keywords. We will also integrate a component for analyzing the sentiment structure of the profiles. We believe that the findings from our work on applying IR techniques to the task of Electronic Career Guidance generalize to similar application domains, where topics and documents display similar properties (with respect to their length, free-text structure and mismatch of vocabularies) and domain and lexical knowledge is required to achieve high levels of performance.

Acknowledgments

This work was supported by the German Research Foundation under grant "Semantic Information Retrieval from Texts in the Example Domain Electronic Career Guidance", GU 798/1-2. We are grateful to the *Bundesagentur für Arbeit* for providing the BERUFEnet corpus. We would like to thank the anonymous reviewers for valuable feedback on this paper. We would also like to thank Piklu Gupta for helpful comments.

References

David Ahn, Valentin Jijkoun, Gilad Mishne, Karin Müller, Maarten de Rijke, and Stefan Schlobach. 2004. Using Wikipedia at the TREC QA Track. In *Proceedings of TREC 2004*.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating WordNet-based Measures of Semantic Distance. *Computational Linguistics*, 32(1).

Razvan Bunescu and Marius Pasca. 2006. Using Encyclopedic Knowledge for Named Entity Disambiguation. In *Proceedings of ACL*, pages 9–16, Trento, Italy.

Christiane Fellbaum. 1998. *WordNet An Electronic Lexical Database*. MIT Press, Cambridge, MA.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. In *Proceedings of The 20th International Joint Conference on Artificial Intelligence (IJCAI)*, Hyderabad, India, January.

Julio Gonzalo, Felisa Verdejo, Irina Chugur, and Juan Cigarran. 1998. Indexing with WordNet synsets can improve text retrieval. In *Proceedings of the Coling-ACL '98 Workshop Usage of WordNet in Natural Language Processing Systems,* Montreal, Canada, August.

Iryna Gurevych and Hendrik Niederlich. 2005. Computing semantic relatedness in german with revised information content metrics. In *Proceedings of "OntoLex 2005 - Ontologies and Lexical Resources" IJCNLP'05 Workshop*, pages 28–33, October 11 – 13.

Iryna Gurevych. 2005. Using the Structure of a Conceptual Network in Computing Semantic Relatedness. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing*, pages 767–778, Jeju Island, Republic of Korea.

Mario Jarmasz and Stan Szpakowicz. 2003. Roget's thesaurus and semantic similarity. In *RANLP*, pages 111–120.

Boris Katz, Gregory Marton, Gary Borchardt, Alexis Brownell, Sue Felshin, Daniel Loreto, Jesse Louis-Rosenberg, Ben Lu, Federico Mora, Stephan Stiller, Ozlem Uzuner, and Angela Wilcox. 2005. External knowledge sources for question answering. In *Proceedings of the 14th Annual Text REtrieval Conference (TREC'2005)*, November.

Claudia Kunze, 2004. *Lexikalisch-semantische Wortnetze*, chapter Computerlinguistik und Sprachtechnologie, pages 423–431. Spektrum Akademischer Verlag.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, pages 296–304. Morgan Kaufmann, San Francisco, CA.

Dan Moldovan and Rada Mihalcea. 2000. Using WordNet and lexical operators to improve Internet searches. *IEEE Internet Computing*, 4(1):34–43.

Jane Morris and Graeme Hirst. 2004. Non-Classical Lexical Semantic Relations. In *Workshop on Computational Lexical Semantics, Human Language Technology Conference of the North American Chapter of the ACL*, Boston.

Christof Müller and Iryna Gurevych. 2006. Exploring the Potential of Semantic Relatedness in Information Retrieval. In *Proceedings of LWA 2006 Lernen - Wissensentdeckung - Adaptivität: Information Retrieval*, pages 126–131, Hildesheim, Germany. GI-Fachgruppe Information Retrieval.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual Correlates of Synonymy. *Communications of the ACM*, 8(10):627–633.

Gerard Salton and Michael J. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York.

Gerard Salton, Edward Fox, and Harry Wu. 1983. Extended boolean information retrieval. *Communication of the ACM*, 26(11):1022–1036.

Alan F. Smeaton, Fergus Kelledy, and Ruari O'Donell. 1994. TREC-4 Experiments at Dublin City University: Thresholding posting lists, query expansion with WordNet and POS tagging of Spanish. In *Proceedings of TREC-4*, pages 373–390.

Peter D. Turney. 2006. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.

Ellen Vorhees. 1994. Query expansion using lexical-semantic relations. In *Proceedings of the 17th Annual ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 61–69.

DeWitt Wallace and Lila Acheson Wallace. 2005. *Reader's Digest, das Beste für Deutschland*. Jan 2001–Dec 2005. Verlag Das Beste, Stuttgart.

Torsten Zesch and Iryna Gurevych. 2006. Automatically Creating Datasets for Measures of Semantic Relatedness. In *Proceedings of the Workshop on Linguistic Distances*, pages 16–24, Sydney, Australia, July. Association for Computational Linguistics.

Torsten Zesch, Iryna Gurevych, and Max Mühlhäuser. 2007a. Analyzing and Accessing Wikipedia as a Lexical Semantic Resource. In *Biannual Conference of the Society for Computational Linguistics and Language Technology*, pages 213–221, Tuebingen, Germany.

Torsten Zesch, Iryna Gurevych, and Max Mühlhäuser. 2007b. Comparing Wikipedia and German Wordnet by Evaluating Semantic Relatedness on Multiple Datasets. In *Proceedings of NAACL-HLT*.

Extracting Social Networks and Biographical Facts From Conversational Speech Transcripts

Hongyan Jing
IBM T.J. Watson Research Center
1101 Kitchawan Road
Yorktown Heights, NY 10598
hjing@us.ibm.com

Nanda Kambhatla
IBM India Research Lab
EGL, Domlur Ring Road
Bangalore - 560071, India
kambhatla@in.ibm.com

Salim Roukos
IBM T.J. Watson Research Center
1101 Kitchawan Road
Yorktown Heights, NY 10598
roukos@us.ibm.com

Abstract

We present a general framework for automatically extracting social networks and biographical facts from conversational speech. Our approach relies on fusing the output produced by multiple information extraction modules, including entity recognition and detection, relation detection, and event detection modules. We describe the specific features and algorithmic refinements effective for conversational speech. These cumulatively increase the performance of social network extraction from 0.06 to 0.30 for the development set, and from 0.06 to 0.28 for the test set, as measured by f-measure on the ties within a network. The same framework can be applied to other genres of text — we have built an automatic biography generation system for general domain text using the same approach.

1 Introduction

A social network represents social relationships between individuals or organizations. It consists of *nodes* and *ties*. *Nodes* are individual actors within the networks, generally a person or an organization. *Ties* are the relationships between the nodes. Social network analysis has become a key technique in many disciplines, including modern sociology and information science.

In this paper, we present our system for automatically extracting social networks and biographical facts from conversational speech transcripts by integrating the output of different IE modules. The IE modules are the building blocks; the fusing module depicts the ways of assembling these building blocks. The final output depends on which fundamental IE modules are used and how their results are integrated.

The contributions of this work are two fold. We propose a general framework for extracting social networks and biographies from text that applies to conversational speech as well as other genres, including general newswire stories. Secondly, we present specific methods that proved effective for us for improving the performance of IE systems on conversational speech transcripts. These improvements include feature engineering and algorithmic revisions that led to a nearly five-fold performance increase for both development and test sets.

In the next section, we present our framework for extracting social networks and other biographical facts from text. In Section 3, we discuss the refinements we made to our IE modules in order to reliably extract information from conversational speech transcripts. In Section 4, we describe the experiments, evaluation metrics, and the results of social network and biography extraction. In Section 5, we show the results of applying the framework to other genres of text. Finally, we discuss related work and conclude with lessons learned and future work.

2 The General Framework

For extraction of social networks and biographical facts, our approach relies on three standard IE modules — entity detection and recognition, relation detection, and event detection — and a fusion module that integrates the output from the three IE systems.

2.1 Entity, Relation, and Event Detection

We use the term *entity* to refer to a person, an organization, or other real world entities, as adopted

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 1040–1047,
Prague, Czech Republic, June 2007.

in the Automatic Content Extraction (ACE) Workshops (ACE, 2005). A *mention* is a reference to a real world entity. It can be *named* (e.g. "John Lennon"), *nominal* (e.g. "mother"), or *pronominal* (e.g. "she").

Entity detection is generally accomplished in two steps: first, a mention detection module identifies all the mentions of interest; second, a co-reference module merges mentions that refer to the same entity into a single co-reference chain.

A *relation detection* system identifies (typically) binary relationships between pairs of mentions. For instance, for the sentence "I'm in New York", the following relation exists: *locatedAt (I, New York)*.

An *event detection* system identifies events of interest and the arguments of the event. For example, from the sentence "John married Eva in 1940", the system should identify the marriage event, the people who got married and the time of the event.

The latest ACE evaluations involve all of the above tasks. However, as shown in the next section, our focus is quite different from ACE — we are particularly interested in improving performance for conversational speech and building on top of ACE tasks to produce social networks and biographies.

2.2 Fusion Module

The fusion module merges the output from IE modules to extract social networks and biographical facts. For example, if a relation detection system has identified the relation *motherOf (mother, my)* from the input sentence "my mother is a cook", and if an entity recognition module has generated entities referenced by the mentions {*my, Josh, me, I, I, ...*} and {*mother, she, her, her, Rosa.....*}, then by replacing *my* and *mother* with the named mentions within the same co-reference chains, the fusion module produces the following nodes and ties in a social network: *motherOf (Rosa, Josh)*.

We generate the nodes of social networks by selecting all the PERSON entities produced by the entity recognition system. Typically, we only include entities that contain at least one *named* mention. To identify ties between nodes, we retrieve all relations that indicate social relationships between a pair of nodes in the network.

We extract biographical profiles by selecting the events (extracted by the event extraction module) and corresponding relations (extracted by the relation extraction module) that involve a given individual as an argument. When multiple documents are used, then we employ a cross-document co-reference system.

3 Improving Performance for Conversational Speech Transcripts

Extracting information from conversational speech transcripts is uniquely challenging. In this section, we describe the data collection used in our experiments, and explain specific techniques we used to improve IE performance on this data.

3.1 Conversational Speech Collection

We use a corpus of videotaped, digitized oral interviews with Holocaust survivors in our experiments. This data was collected by the USC Shoah Foundation Institute (formerly known as the Visual History Foundation), and has been used in many research activities under the Multilingual Access to Large Spoken Archives (MALACH) project (Gustman et al., 2002; Oard et al., 2004). The collection contains oral interviews in 32 languages from 52,000 survivors, liberators, rescuers and witnesses of the Holocaust.

This data is very challenging. Besides the usual characteristics of conversational speech, such as speaker turns and speech repairs, the interview transcripts contain a large percentage of ungrammatical, incoherent, or even incomprehensible clauses (a sample interview segment is shown in Figure 1). In addition, each interview covers many people and places over a long time period, which makes it even more difficult to extract social networks and biographical facts.

> *speaker2* in on that ninth of November nineteen hundred thirty eight I was with my parents at home we heard not through the we heard even through the windows the crashing of glass the crashing of and and they are our can't

Figure 1: Sample interview segment.

3.2 The Importance of Co-reference Resolution

Our initial attempts at social network extraction for the above data set resulted in a very poor score

of 0.06 f-measure for finding the relations within a network (as shown in Table 3 as baseline performance).

An error analysis indicated poor co-reference resolution to be the chief culprit for the low performance. For instance, suppose we have two clauses: "his mother's name is Mary" and "his brother Mark went to the army". Further suppose that "his" in the first clause refers to a person named "John" and "his" in the second clause refers to a person named "Tim". If the co-reference system works perfectly, the system should find a social network involving four people: {*John, Tim, Mary, Mark*}, and the ties: *motherOf (Mary, John)*, and *brotherOf (Mark, Tim)*. However, if the co-reference system mistakenly links "John" to "his" in the second clause and links "Tim" to "his" in the first clause, then we will still have a network with four people, but the ties will be: *motherOf (Mary, Tim)*, and *brotherOf (Mark, John)*, which are completely wrong. This example shows that co-reference errors involving mentions that are relation arguments can lead to very bad performance in social network extraction.

Our existing co-reference module is a state-of-the-art system that produces very competitive results compared to other existing systems (Luo et al., 2004). It traverses the document from left to right and uses a mention-synchronous approach to decide whether a mention should be merged with an existing entity or start a new entity.

However, our existing system has shortcomings for this data: the system lacks features for handling conversational speech, and the system often makes mistakes in pronoun resolution. Resolving pronominal references is very important for extracting social networks from conversational speech, as illustrated in the previous example.

3.3 Improving Co-reference for Conversational Speech

We developed a new co-reference resolution system for conversational speech transcripts. Similar to many previous works on co-reference (Ng, 2005), we cast the problem as a classification task and solve it in two steps: (1) train a classifier to determine whether two mentions are co-referent or not, and (2) use a clustering algorithm to partition the mentions into clusters, based on the pairwise predictions.

We added many features to our model specifically designed for conversational speech, and significantly improved the agglomerative clustering used for co-reference, including integrating relations as constraints, and designing better cluster linkage methods and clustering stopping criteria.

3.3.1 Adding Features for Conversational Speech

We added many features to our model specifically designed for conversational speech:

Speaker role identification. In manual transcripts, the speaker turns are given and each speaker is labeled differently (e.g. *"speaker1"*, *"speaker2"*), but the identity of the speaker is not given. An interview typically involves 2 or more speakers and it is useful to identify the roles of each speaker (e.g. interviewer, interviewee, etc.). For instance, "you" spoken by the interviewer is likely to be linked with "I" spoken by the interviewee, but "you" spoken by the third person in the interview is more likely to be referring to the interviewer than to the interviewee.

We developed a program to identify the speaker roles. The program classifies the speakers into three categories: interviewer, interviewee, and others. The algorithm relies on three indicators — number of turns by each speaker, difference in number of words spoken by each speaker, and the ratio of first-person pronouns such as "I", "me", and "we" vs. second-person pronouns such as "you" and "your". This speaker role identification program works very well when we checked the results on the development and test set — the interviewers and survivors in all the documents in the development set were correctly identified.

Speaker turns. Using the results from the speaker role identification program, we enrich certain features with speaker turn information. For example, without this information, the system cannot distinguish "I" spoken by an interviewer from "I" spoken by an interviewee.

Spelling features for speech transcripts. We add additional spelling features so that mentions such as "Cyla C Y L A Lewin" and "Cyla Lewin" are considered as exact matches. Names with spelled-out letters occur frequently in our data collection.

Name Patterns. We add some features that capture frequent syntactic structures that speakers use to express names, such as "her name is Irene", "my cousin Mark", and "interviewer Ellen".

Pronoun features. To improve the perfor-

mance on pronouns, we add features such as the speaker turns of the pronouns, whether the two pronouns agree in person and number, whether there exist other mentions between them, etc.

Other miscellaneous features. We also include other features such as gender, token distance, sentence distance, and mention distance.

We trained a maximum-entropy classifier using these features. For each pair of mentions, the classifier outputs the probability that the two mentions are co-referent.

We also modified existing features to make them more applicable to conversational speech. For instance, we added pronoun-distance features taking into account the presence of other pronominal references in between (if so, the types of the pronouns), other mentions in between, etc.

3.3.2 Improving Agglomerative Clustering

We use an agglomerative clustering approach for partitioning mentions into entities. This is a bottom-up approach which joins the closest pair of clusters (i.e., entities) first. Initially, each mention is placed into its own cluster. If we have N mentions to cluster, we start with N clusters.

The intuition behind choosing the agglomerative method is to merge the most confident pairs first, and use the properties of existing clusters to constrain future clustering. This seems to be especially important for our data collection, since conversational speech tends to have a lot of repetitions or local structures that indicate co-reference. In such cases, it is beneficial to merge these closely related mentions first.

Cluster linkage method. In agglomerative clustering, each cycle merges two clusters into a single cluster, thus reducing the number of clusters by one. We need to decide upon a method of measuring the distance between two clusters.

At each cycle, the two mentions with the highest co-referent probability are linked first. This results in the merging of the two clusters that contain these two mentions.

We improve upon this method by imposing *minimal distance criteria* between clusters. Two clusters C_1 and C_2 can be combined only if the distance between all the mentions from C_1 and all the mentions from C_2 is above the minimal distance threshold. For instance, suppose $C_1 = \{he, father\}$, and $C_2 = \{he, brother\}$, and "he" from C_1 and "he" from C_2 has the highest linkage probability. The standard single linkage method will combine these two clusters, despite the fact that "father" and "brother" are very unlikely to be linked. Imposing minimal distance criteria can solve this problem and prevent the linkage of clusters which contain very dissimilar mentions. In practice, we used multiple minimal distance thresholds, such as minimal distance between two named mentions and minimal distance between two nominal mentions.

We chose not to use complete or average linkage methods. In our data collection, the narrations contain a lot of pronouns and the focus tends to be very local. Whereas the similarity model may be reasonably good at predicting the distance between two pronouns that are close to each other, it is not good at predicting the distance between pronouns that are furthur apart. Therefore, it seems more reasonable to use single linkage method with modifications than complete or average linkage methods.

Using relations to constrain clustering. Another novelty of our co-reference system is the use of relations for constraining co-reference. The idea is that two clusters should not be merged if such merging will introduce contradictory relations. For instance, if we know that person entity A is the mother of person entity B, and person entity C is the sister of B, then A and C should not be linked since the resulting entity will be both the mother and the sister of B.

We construct co-existent relation sets from the training data. For any two pairs of entities, we collect all the types of relations that exist between them. These types of relations are labeled as co-existent. For instance, "motherOf" and "parentOf" can co-exist, but "motherOf" and "sisterOf" cannot. By using these relation constraints, the system refrains from generating contradictory relations in social networks.

Speed improvement. Suppose the number of mentions is N, the time complexity of simple linkage method is $O(N^2)$. With the minimal distance criteria, the complexity is $O(N^3)$. However, N can be dramatically reduced for conversational transcripts by first linking all the first-person pronouns by each speaker.

4 Experiments

In this section, we describe the experimental setup and present sample outputs and evaluation results.

	Train	Dev	Test
Words	198k	73k	255k
Mentions	43k	16k	56k
Relations	7K	3k	8k

Table 2: Experimental Data Sets.

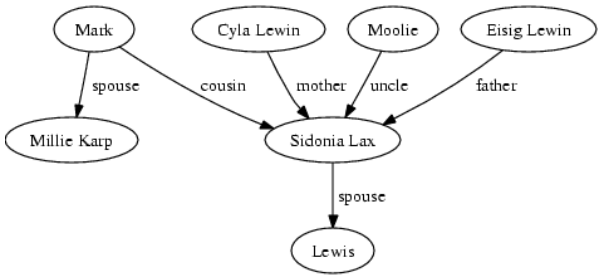


Figure 2: Social network extracted by the system.

4.1 Data Annotation

The data used in our experiments consist of partial or complete English interviews of Holocaust survivors. The input to our system is transcripts of interviews.

We manually annotated manual transcripts with entities, relations, and event categories, specifically designed for this task and the results of careful data analysis. The annotation was performed by a single annotator over a few months. The annotation categories for entities, events, and relations are shown in Table 1. Please note that the event and relation definitions are slightly different than the definitions in ACE.

4.2 Training and Test Sets

We divided the data into training, development, and test data sets. Table 2 shows the size of each data set. The training set includes transcripts of partial interviews. The development set consists of 5 complete interviews, and the test set consists of 15 complete interviews. The reason that the training set contains only partial interviews is due to the high cost of transcription and annotation. Since those partial interviews had already been transcribed for speech recognition purpose, we decided to reuse them in our annotation. In addition, we transcribed and annotated 20 complete interviews (each interview is about 2 hours) for building the development and test sets, in order to give a more accurate assessment of extraction performance.

4.3 Implementation

We developed the initial entity detection, relation detection, and event detection systems using the same techniques as our submission systems to ACE (Florian et al., 2004). Our submission systems use statistical approaches, and have ranked in the top tier in ACE evaluations. We easily built the models for our application by retraining existing systems with our training set.

The entity detection task is accomplished in two steps: mention detection and co-reference resolution. The mention detection is formulated as a labeling problem, and a maximum-entropy classifier is trained to identify all the mentions.

Similarly, relation detection is also cast as a classification problem — for each pair of mentions, the system decides which type of relation exists between them. It uses a maximum-entropy classifier and various lexical, contextual, and syntactic features for such predications.

Event detection is accomplished in two steps: first, identifying the event anchor words using an approach similar to mention detection; then, identifying event arguments using an approach similar to relation detection.

The co-reference resolution system for conversational speech and the fusion module were developed anew.

4.4 The Output

The system aims to extract the following types of information:

- The social network of the survivor.
- Important biographical facts about each person in the social network.
- Track the movements of the survivor and other individuals in the social network.

Figure 2 shows a sample social network extracted by the system (only partial of the network is shown). Figure 3 shows sample biographical facts and movement summaries extracted by the system. In general, we focus more on higher precision than recall.

4.5 Evaluation

In this paper, we focus only on the evaluation of social network extraction. We first describe the metrics for social network evaluation and then present the results of the system.

Entity (12)	Event (8)	Relation (34)		
		Social Rels (12)	Event Args (8)	Bio Facts (14)
AGE	CUSTODY	aidgiverOf	affectedBy	bornAt
COUNTRY	DEATH	auntOf	agentOf	bornOn
DATE	HIDING	cousinOf	participantIn	citizenOf
DATEREF	LIBERATION	fatherOf	timeOf	diedAt
DURATION	MARRIAGE	friendOf	travelArranger	diedOn
GHETTOORCAMP	MIGRATION	grandparentOf	travelFrom	employeeOf
OCCUPATION	SURVIVAL	motherOf	travelPerson	hasProperty
ORGANIZATION	VIOLENCE	otherRelativeOf	travelTo	locatedAt
OTHERLOC		parentOf		managerOf
PEOPLE		siblingOf		memberOf
PERSON		spouseOf		near
SALUTATION		uncleOf		partOf
				partOfMany
				resideIn

Table 1: Annotation Categories for Entities, Events, and Relations.

Sidonia Lax:
date of birth: June the eighth nineteen twenty seven

Movements:
Moved To: Auschwitz
Moved To: United States
... ...

Figure 3: Biographical facts and movement summaries extracted by the system.

To compare two social networks, we first need to match the *nodes* and *ties* between the networks. Two nodes (i.e., entities) are matched if they have the same canonical name. Two ties (i.e., edges or relations) are matched if these three criteria are met: they contain the same type of relations, the arguments of the relation are the same, and the order of the arguments are the same if the relation is unsymmetrical.

We define the the following measurements for social network evaluation: the *precision for nodes (or ties)* is the ratio of common nodes (or ties) in the two networks to the total number of nodes (or ties) in the system output, the *recall for nodes (or ties)* is the ratio of common nodes (or ties) in the two networks to the total number of nodes/ties in the reference output, and the *f-measure for nodes (or ties)* is the harmonic mean of precision and recall for nodes (or ties). The *f-measure for ties* indicates the overall performance of social network extraction.

F-mea	Dev		Test	
	Baseline	New	Baseline	New
Nodes	0.59	0.64	0.62	0.66
Ties	0.06	**0.30**	0.06	**0.28**

Table 3: Performance of social network extraction.

Table 3 shows the results of social network extraction. The new co-reference approach improves the performance for f-measure on ties by five-fold on development set and by nearly five-fold for test set.

We also tested the system using automatic transcripts by our speech recognition system. Not surprisingly, the result is much worse: the nodes f-measure is 0.11 for the test set, and the system did not find any relations. A few factors are accountable for this low performance: (1) Speech recognition is very challenging for this data set, since the testimonies contained elderly, emotional, accented speech. Given that the speech recognition system fails to recognize most of the person names, extraction of social networks is difficult. (2) The extraction systems perform worse on automatic transcripts, due to the quality of the automatic transcript, and the discrepancy between the training and test data. (3) Our measurements are very strict, and no partial credit is given to partially correct entities or relations.

We decided not to present the evaluation results of the individual components since the performance of individual components are not at all indicative of the overall performance. For instance, a single pronoun co-reference error might slighlty

change the co-reference score, but can introduce a serious error in the social network, as shown in the example in Section 3.2.

5 Biography Generation from General Domain Text

We have applied the same framework to biography generation from general news articles. This general system also contains three fundamental IE systems and a fusion module, similar to the work presented in the paper. The difference is that the IE systems are trained on general news text using different categories of entities, relations, and events.

A sample biography output extracted from TDT5 English documents is shown in Figure 4. The numbers in brackets indicate the corpus count of the facts.

Saddam Hussein:
Basic Information:
citizenship: Iraq [203]
occupation: president [4412], leader [1792], dictator [664],...
relative: odai [89], qusay [65], uday [65],...

Life Events:
places been to: bagdad [403], iraq [270], palaces [149]...
Organizations associated with: manager of baath party [1000], ...
Custody Events: Saddam was arrested [52],
Communication Events: Saddam said [3587]
... ...

Figure 4: Sample biography output.

6 Related Work

While there has been previous work on extracting social networks from emails and the web (Culotta et al., 2004), we believe this is the first paper to present a full-fledged system for extracting social networks from conversational speech transcripts.

Similarly, most of the work on co-reference resolution has not focused on conversational speech. (Ji et al., 2005) uses semantic relations to refine co-reference decisions, but in a approach different from ours.

7 Conclusions and Future Work

We have described a novel approach for extracting social networks, biographical facts, and movement summaries from transcripts of oral interviews with Holocaust survivors. We have improved the performance of social network extraction five-fold, compared to a baseline system that already uses state-of-the-art technology. In particular, we improved the performance of co-reference resolution for conversational speech, by feature engineering and improving the clustering algorithm.

Although our application data consists of conversational speech transcripts in this paper, the same extraction approach can be applied to general-domain text as well. Extracting general, rich social networks is very important in many applications, since it provides the knowledge of who is connected to whom and how they are connected.

There are many interesting issues involved in biography generation from a large data collection, such as how to resolve contradictions. The counts from the corpus certainly help to filter out false information which would otherwise be difficult to filter. But better technology at detecting and resolving contradictions will definitely be beneficial.

Acknowledgment

We would like to thank Martin Franz and Bhuvana Ramabhadran for their help during this project. This project is funded by NSF under the Information Technology Research (ITR) program, NSF IIS Award No. 0122466. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

2005. Automatic content extraction. http://www.nist.gov/speech/tests/ace/.

Aron Culotta, Ron Bekkerman, and Andrew McCallum. 2004. Extracting social networks and contact information from email and the web. In *CEAS*, Mountain View, CA.

Radu Florian, Hany Hassan, Abraham Ittycheriah, Hongyan Jing, Nanda Kambhatla, Xiaoqiang Luo, Nicolas Nicolov, and Salim Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *Proceedings of. HLT-NAACL 2004*.

Samuel Gustman, Dagobert Soergeland Douglas Oard, William Byrne, Michael Picheny, Bhuvana Ramabhadran, and Douglas Greenberg. 2002. Supporting access to large digital oral history archives. In *Proceedings of the Joint Conference on Digital Libraries*, pages 18–27.

Heng Ji, David Westbrook, and Ralph Grishman. 2005. Using semantic relations to refine coreference decisions. In *Proceedings of HLT/EMNLP'05*, Vancouver, B.C., Canada.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL2004)*, pages 135–142, Barcelona, Spain.

Vincent Ng. 2005. Machine learning for coreference resolution: From local classification to global ranking. In *Proceedings of ACL'04*.

D. Oard, D. Soergel, D. Doermann, X. Huang, G.C. Murray, J. Wang, B. Ramabhadran, M. Franz, S. Gustman, J. Mayfield, L. Kharevych, and S. Strassel. 2004. Building an information retrieval test collection for spontaneous conversational speech. In *Proceedings of SIGIR'04*, Sheffield, U.K.

Author Index

ISBN 978-1-932432-86-2