HLT-NAACL 2006

Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics

Proceedings of the Main Conference

Robert C. Moore, General Chair
Jeff Bilmes, Jennifer Chu-Carroll and Mark Sanderson
Program Committee Chairs

June 4-9, 2006
New York, New York, USA

Published by the Association for Computational Linguistics
http://www.aclweb.org

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

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

Preface from the General Chair

This year marks the third time that the conference on Human Language Technology has combined with the North American chapter meeting of the Association for Computational Linguistics. The roster of accepted papers reveals an eclectic mix of topics in natural-language processing, speech processing, and information retrieval. A gratifying number of the papers are difficult to classify because they span more than one of these three major areas of human language technology. For example, the boundary between natural-language processing and information retrieval is hard to draw in the papers that focus on the World Wide Web as a corpus; moreover, several of these include speech-related aspects as well.

The crazy thing about putting on a conference like this is that you start out with a group of people who have never done it before, and by the time they really figure out what they are doing, the conference is over and you replace them with another group of people who have never done it before! To do a good job as general chair, however, there is only one really important thing to learn: pick really good people to do all the other jobs, sit back, and let them do all the work. I have been very fortunate to have a great group of conference organizers to rely on: the NYU local arrangements committee, headed by Satoshi Sekine; the program chairs Jennifer Chu-Carroll, Jeff Bilmes, and Mark Sanderson; the demonstration chairs Alex Rudnicky, John Dowding, and Natasa Milic-Frayling; the publications chairs Sanjeev Khudanpur and Brian Roark; the publicity chairs Dan Gildea, Ciprian Chelba, and Eric Brown; the sponsorship and exhibits chairs Ed Hovy and Patrick Pantel; the tutorial chairs Chris Manning, Doug Oard, and Jim Glass; the workshop chairs Lucy Vanderwende, Roberto Pieraccini, and Liz Liddy; the Doctoral Consortium chairs Matt Huenerfauth and Bo Pang, and their faculty advisor, Mitch Marcus.

I would also like to thank ACL Business Manager Priscilla Rasmussen, who took on even more responsibility than she usually does to insure that the conference is a success; and the NAACL executive committee and HLT advisory board for encouragement and advice when we were just getting started and didn't know much about what needed to be done. Finally, I would like to thank the senior program committee members, all the paper reviewers, the student volunteers, and the conference sponsors, without whom the conference could not happen.

Robert C. Moore
Microsoft Research
General Chair

Preface from the Program Co-Chairs

It is with pleasure that we preface the publications of the 2006 *Human Language Technology conference — North American chapter of the Association for Computational Linguistics annual meeting (HLT-NAACL 2006)*. The conference has a number of formats by which refereed work can be presented: full papers, short papers (either as a talk or poster), and demonstrations. As befits this multi-disciplinary conference, papers were submitted across the three topics of computational linguistics, information retrieval and speech recognition. This year, 257 full papers were submitted and 62 accepted (25% acceptance rate), 127 short papers submitted and 52 accepted (41% rate). It is pleasing to report that these numbers mark a strong increase in submissions compared to the last HLT NAACL conference run in 2004.

The selection of the high quality submissions in these proceedings was the product of a two tiered reviewing system. The three PC chairs selected 28 senior program committee (PC) members, who are internationally recognized for their subject expertise. This group constituted the top tier of the PC. Each of the members selected a group of reviewers to review both the full and short submitted papers. The complete PC numbered around 250. Three reviewers and one senior PC person were assigned per paper. Reviewing was double blinded. The senior PC oversaw the reviewing process, helped resolve any disputes, and at the end produced, for each paper, an overview of the reviewers' comments along with a preliminary decision on whether the submission should be accepted or not. These decisions formed the basis of discussion at a program committee meeting. Separate PC meetings were held for full and short papers. For full, a one day meeting was held at IBM Research Watson, NY; for short papers, a telephone conference call was held between the three PC chairs.

The senior PC also nominated candidates for best paper and best student paper, the two selected for the prizes were chosen by the PC chairs working in conjunction with the senior PCs. The papers that won were "Probabilistic Context-Free Grammar Induction Based on Structural Zeros" by Mehryar Mohri and Brian Roark and "Prototype-Driven Learning for Sequence Models" by Aria Haghighi and Dan Klein. Congratulations to them both.

We are indebted to all those who submitted papers to the conference and to all the reviewers and senior PC members who volunteered their time to help us in the selection process for the conference. We are particularly indebted to all the senior PC members who attended the PC meeting in January and found funds to pay for themselves to attend the meeting. Thanks guys, that was particularly generous of you. We are also grateful to IBM Watson for providing facilities for the PC meeting, Bob Moore for all of his prompt advice and help and a final thanks to Rich Gerber who ran and helped modify the START reviewing system

The HLT-NAACL conference has a PC chair for each of its three disciplines. Although work tasks were shared between the three chairs equally, as computational linguistics received by far the greatest number of submissions, Jennifer Chu-Carroll ended up having to oversee more papers and recruit more senior PC members than the other two chairs, she also volunteered to host the PC meeting at IBM. Therefore, the two other chairs of HLT-NAACL 2006 (Mark Sanderson & Jeff Bilmes), wish to thank Jennifer for all of her additional work in pulling this conference together. Jennifer, it couldn't have been done without you.

Jennifer Chu-Carroll — IBM Research (Watson)
Jeff Bilmes — University of Washington
Mark Sanderson — University of Sheffield
Program Co-Chairs

Conference Organizers

General Chair:

Robert C. Moore - Microsoft Research (Redmond)

Local Arrangements Committee:

Satoshi Sekine, NYU (chair)
Ralph Grishman, NYU (co-chair)
Koji Murakami, NYU (webmaster)
David Westbrook, NYU (associate webmaster)
Adam Meyers, NYU (volunteer coordinator)

Program Committee Chairs:

Jeff Bilmes, University of Washington
Jennifer Chu-Carroll, IBM T.J. Watson Research Center
Mark Sanderson, University of Sheffield

ACL office:

Priscilla Rasmussen

Demonstration Chairs:

Alex Rudnicky, Carnegie Mellon University
John Dowding, University of California, Santa Cruz
Natasa Milic-Frayling, Microsoft Research (Cambridge)

Publications Chairs:

Sanjeev Khudanpur, Johns Hopkins University
Brian Roark, OGI-Oregon Health & Science University

Publicity Chairs:

Dan Gildea, University of Rochester
Ciprian Chelba, Microsoft Research (Redmond)
Eric Brown, IBM Research (Watson)

Sponsorship and Exhibits Chairs:

Ed Hovy, USC-ISI
Patrick Pantel, USC-ISI

Tutorial Chairs:

Chris Manning, Stanford University
Doug Oard, University of Maryland
Jim Glass, MIT

Workshop Chairs:

Lucy Vanderwende, Microsoft Research (Redmond)
Roberto Pieraccini, Tell-Eureka
Liz Liddy, Syracuse University

Doctoral Consortium Chairs:

Matt Huenerfauth, University of Pennsylvania
Bo Pang, Cornell University
Mitch Marcus, University of Pennsylvania (Faculty advisor)

Human Language Technology Advisory Board:

Donna Harman, NIST (2005-06)
Mary Harper, University of Maryland (2005-06)
Julia Hirschberg, Columbia University (2005)
Graeme Hirst, University of Toronto (2005-06)
Sanjeev Khudanpur, Johns Hopkins University (2005-06)
Tatiana Korelsky, NSF (2006)
Raymond Mooney, University of Texas at Austin (2005-06)
Robert Moore, Microsoft Research (2005-06)
Heather McCallum-Bayliss, ARDA (2006)
Joseph Olive, DARPA (2006)
John Prange, ARDA (2005)
Dragomir Radev, University of Michigan (2005)
Owen Rambow, Columbia University (2006)
Ellen Riloff, University of Utah (2005)
Charles Wayne, DARPA (2005)

Senior Program Committee Members:

Johan Bos, University of Roma "La Sapienza"
Jamie Callan, Carnegie Mellon University
Joyce Chai, Michigan State University
Jason Eisner, Johns Hopkins University
Mark Gales, Cambridge University
Fredric Gey, University of California Berkeley
Roxana Girju, UIUC
Mark Hasegawa-Johnson, UIUC
Julia Hirschberg, Columbia University
Alon Lavie, Carnegie Mellon University
Wei-Ying Ma, Microsoft Research Asia
Mehryar Mohri, Courant Institute/Google
Marius Pasca, Google
Gerald Penn, University of Toronto
Dragomir Radev, University of Michigan
Owen Rambow, CCLS, Columbia University
Steve Renals, University of Edinburgh
Stefan Riezler, Google
Rohini Srihari, SUNY Buffalo
Amanda Stent, SUNY Stony Brook
Michael Strube, EML Research
Christoph Tillmann, IBM T.J. Watson Research Center
Peter Turney, National Research Council Canada
Ellen Voorhees, NIST
Ralph Weischedel, BBN Technologies
Fei Xia, University of Washington
ChengXiang Zhai, UIUC
Ming Zhou, Microsoft Research Asia

Program Committee Members:

Steven Abney, U. of Michigan
Alex Acero, Microsoft Research
Cyril Allauzen, Courant Institute
Yaser Al-Onaizan, IBM
Abeer Alwan, UCLA
Elisabeth Andre, U. Augsburg
Chinatsu Aone, SRA
Doug Appelt, SRI International
Michiel Bacchiani, Google Inc
Tim Baldwin, U. of Melbourne
Srinivas Bangalore, AT&T Labs – Research
Regina Barzilay, MIT CSAIL
John Bateman, U. Bremen
Jerome Bellegarda, Apple Computer, Inc.
Anja Belz, ITRI, U. of Brighton
Pushpak Bhattacharya, Indian Inst. of Technology
Timothy Bickmore, Northeastern U.
Patrick Blackburn, INRIA Lorraine
Eric Brown, IBM
Ralf Brown, Carnegie Mellon U.
John Burger, MITRE
Bill Byrne, U. of Cambridge
Donna Byron, Ohio State U.
Charles Callaway, U. of Edinburgh
Chris Callison-Burch, U. of Edinburgh
Giuseppe Carenini, U. of British Columbia
Rolf Carlson, KTH
Violetta Cavalli-Sforza, San Francisco State U.
Ciprian Chelba, Google
John Chen, Janya Inc.
Stanley Chen, IBM
David Chiang, USC-ISI
Lee-Feng Chien, Academia Sinica
Tat-Seng Chua, National U. of Singapore
Grace Chung, MIT
Alexander Clark, Royal Holloway U. of London
Stephen Clark, Oxford U.
Michael Collins, MIT CSAIL
Tom Cornell, Janya Inc.
Corinna Cortes, Google Research
Cassandre Creswell, Janya Inc.
Mathias Creutz, Helsinki U. of Technology
Dick Crouch, PARC
Ido Dagan, Bar Ilan U.
Tiphaine Dalmas, U. of Edinburgh
Mary Dalrymple, U. of Oxford
Hoa Dang, NIST
Franciska de Jong, U. of Twente
Li Deng, Microsoft
Barbara di Eugenio, U. of Illinois at Chicago
Mona Diab, CCLS, Columbia U.
Bill Dolan, Microsoft Research
Bonnie Dorr, U. of Maryland
Markus Egg, Rijksuniversiteit Groningen
Gunes Erkan, U. of Michigan
Oren Etzioni, U. of Washington
Patrick Fan, Virginia Tech
David Farwell, New Mexico State U.
Eric Fosler-Lussier, Ohio State U.
Anette Frank, DFKI
Robert Frank, Johns Hopkins U.
Bob Frederking, Carnegie Mellon U.
Maria Fuentes, U. Politècnica de Catalunya
Junichi Fukumoto, Ritsumeikan U.
Rob Gaizauskas, U. of Sheffield
Michel Galley, Columbia U.
Jianfeng Gao, Microsoft Research Asia
Claire Gardent, CNRS/LORIA
Daniel Gildea, U. of Rochester
John Goldsmith, U. of Chicago
Sharon Goldwater, Brown U.
Yoshi Gotoh, U. of Sheffield
Mark Greenwood, U. of Sheffield
Ralph Grishman, New York U.
Joakim Gustafson, TeliaSonera
Nizar Habash, CCLS, Columbia U.
Thomas Hain, U. of Sheffield
Keith Hall, Johns Hopkins U.
Susan Haller, SUNY Potsdam
Sanda Harabagiu, U. of Texas at Dallas
Mary Harper, Purdue U.
Anthony Hartley, U. of Leeds
Marti Hearst, U. of California, Berkeley
Peter Heeman, Oregon Graduate Institute
James Henderson, U. of Edinburgh
John Henderson, The MITRE Corporation
Ulf Hermjakob, USC-ISI
Djoerd Hiemstra, U. of Twente

Program Committee Members (continued):

Keikichi Hirose, U. of Tokyo
Jerry Hobbs, USC-ISI
Kristy Hollingshead, Oregon Health & Science U.
Fei Huang, IBM
Diana Inkpen, U. of Ottawa
Martin Jansche, CCLS, Columbia U.
Mark Johnson, Brown U.
Hideo Joho, U. of Glasgow
Gareth Jones, Dublin City U.
Nanda Kambhatla, IBM
Frank Keller, U. of Edinburgh
Kevin Knight, USC-ISI
Philipp Koehn, U. of Edinburgh
Wessel Kraaij, TNO
Jonas Kuhn, Saarland U., Saarbrücken
KL Kwok, City U. of New York
Mirella Lapata, U. of Edinburgh
Victor Lavrenko, U. of Massachusetts at Amherst
Esther Levin, CCNY/CUNY
Gina Levow, U. of Chicago
Jimmy Lin, U. of Maryland
Ken Litkowski, CL Research
Bing Liu, U. of Illinois at Chicago
Andrej Ljolje, AT&T Labs - Research
Bernardo Magnini, ITC-irst
Thomas Mandl, U. Hildesheim
Gideon Mann, U. of Massachusetts at Amherst
Katja Markert, Leeds U.
James Mayfield, JHU/APL
Michael McCord, IBM
Ryan McDonald, U. of Pennsylvania
Helen Meng, Chinese U. of Hong Kong
Teruko Mitamura, Carnegie Mellon U.
Marie-France Moens, Katholieke U. Leuven
Christof Monz, Queen Mary, U. of London
Yukiko Nakano, Tokyo U. of Agriculture & Tech.
Mark-Jan Nederhof, Max Planck Inst. of Psych..
Vincent Ng, U. of Texas at Dallas
Cheng Niu, Microsoft Research Asia
Franz Och, Google
Mari Ostendorf, U. of Washington
Martha Palmer, U. of Colorado
Rebecca Passonneau, CCLS, Columbia U.
Fernando Pereira, U. of Pennsylvania
Graeme Hirst, U. of Toronto
Julia Hockenmaier, U. of Pennsylvania
Chiori Hori, CMU
Rebecca Hwa, U. of Pittsburgh
Abraham Ittycheriah, IBM
Rong Jin, Michigan State U.
Michael Johnston, AT&T Labs - Research
Kristiina Jokinen, U. of Helsinki
Joemon Jose, U. of Glasgow
Min Yen Kan, National U. of Singapore
Kazuaki Kishida, Surugadi U.
Kate Knill, Toshiba Research Europe Ltd
Alexander Koller, U. of the Saarland
Emiel Krahmer, Tilburg U.
Shankar Kumar, Google
Philippe Langlais, U. de Montréal
Alex Lascarides, U. of Edinburgh
Lillian Lee, Cornell U.
Lori Levin, Carnegie Mellon U.
Elizabeth Liddy, Syracuse U.
Chin-Yew Lin, Microsoft Research Asia
Diane Litman, U. of Pittsburgh
Karen Livescu, MIT
Bente Maegaard, U. of Copenhagen
Steve Maiorano, ATP
Inderjeet Mani, MITRE
Daniel Marcu, USC-ISI
Yuji Matsumoto, Nara Inst. of Science and Tech.
Andrew McCallum, U. of Massachusetts at Amherst
Iain McCowan, CSIRO ICT Centre, Australia
Dan Melamed, New York U.
Rada Mihalcea, U. of North Texas
Yusuke Miyao, U. of Tokyo
Dan Moldovan, U. of Texas at Dallas
Isabelle Moulinier, Thompson Legal
Shri Narayanan, USC
Hwee Tou Ng, National U. of Singapore
Jian-Yun Nie, U. of Montréal
Tadashi Nomoto, National Inst. of Japanese Lit.
Mohamed Omar, IBM
Iadh Ounis, U. of Glasgow
Kishore Papineni, IBM
Ted Pedersen, U. of Minnesota, Duluth
Jose Perez-Carballo, California State U., LA

Program Committee Members (continued):

Carol Peters, Italian National Research Council
Richard Power, ITRI, U. of Brighton
John Prager, IBM
Rashmi Prasad, U. of Pennsylvania
Vasin Punyakanok, UIUC
Bhuvana Ramabhadran, IBM
Adwait Ratnaparkhi, Microsoft Research
Ehud Reiter, U. of Aberdeen
Christian Retore', U. Bordeaux 1
Michael Riley, Google, Inc
Alex Rudnicky, Carnegie Mellon U.
Murat Saraçlar, Boğaziçi U.
Michael Schiehlen, U. of Stuttgart
Falk Scholer, RMIT U.
Frank Seide, Microsoft Research Asia
Ben Shahshahani, Yahoo
Candy Sidner, Mitsubishi Electric Research
Frank Soong, Microsoft Research Asia
Mark Steedman, U. of Edinburgh
Suzanne Stevenson, U. of Toronto
Matthew Stone, Rutgers U.
Tomek Strzalkowski, SUNY Albany
Eiichiro Sumita, ATR
Stan Szpakowicz, U. of Ottawa
Joel Tetreault, U. of Pittsburgh, LRDC
David Traum, USC/Inst. of Creative Tech.
Josef van Genabith, Dublin City U.
Lucy Vanderwende, Microsoft Research
Phil Vines, Royal Melbourne Inst. of Tech.
Andy Way, Dublin City U.
Ji-Rong Wen, Microsoft Research Asia
Yoad Winter, Technion, Haifa
Dekai Wu, HKUST
Steve Young, U. of Cambridge
Hugo Zaragoza, Yahoo! Research
Richard Zens, RWTH Aachen U.
Qifeng Zhu, Texas Instruments
Paul Piwek, The Open U.
Sameer Pradhan, BBN Technologies
Kishore Prallahad, Carnegie Mellon U.
Mark Przybocki, NIST
Matthew Purver, CSLI, Stanford U.
Lance Ramshaw, BBN Technologies
Deepak Ravichandran, Google
Norbert Reithinger, DFKI
Steve Richardson, Microsoft Research
Ellen Riloff, U. of Utah
Gregory Sanders, NIST
Anoop Sarkar, Simon Fraser U.
Frank Schilder, Thomson Legal & Regulatory
Sabine Schulte, Saarlandes U.
Stephanie Seneff, MIT CSAIL
Koichi Shinoda, Tokyo Intstitute of Technology
Khalil Sima'an, U. van Amsterdam
Richard Sproat, UIUC
Mark Stevenson, U. of Sheffield
Nicola Stokes, U. of Melbourne
Kristina Streignitz, Northwestern U.
Keh-Yih Su, Behavior Design Corporation
Marc Swerts, Tilburg U.
Egidio Terra, Amazon.com
Kentaro Torisawa, Japan Advanced Inst. of Sci&Tech.
Harald Trost, Medical U. of Vienna
Gertjan van Noord, U. of Groningen
Eric Villemonte de la Clergerie, INRIA
Taro Watanabe, NTT Communication Science Lab
Bonnie Webber, U. of Edinburgh
Janyce Wiebe, U. of Pittsburgh
Christa Womser-Hacker, U. Hildesheim
Roman Yangarber, U. of Helsinki
Deniz Yuret, Koc U.
Dmitry Zelenko, SRA
Yi Zhang, U. of California, Santa Cruz

Additional Reviewers:

Oana Frunza, U. of Ottawa
Michael Gamon, Microsoft Research
Preslav Nakov, U. of Califonia, Berkeley
Carlos Prolo, Pontifícia U. Católica do Rio Grande do Sul
Marcus Sammer, U. of Washington
Marina Sokolova, U. de Montréal
Ana-Maria Popescu, U. of Washington

Table of Contents

Conference Program

Sunday, June 4

9:00–5:30 **Doctoral Consortium**

Tutorials

9:00–12:30 **T1: What's in a Name: Current Methods, Applications, and Evaluation in Multilingual Name Search and Matching**
Sherri Condon and Keith Miller

9:00–12:30 **T2: Beyond EM: Bayesian Techniques for Human Language Technology Researchers**
Hal Daume III

9:00–12:30 **T3: Graph-based Algorithms for Natural Language Processing and Information Retrieval**
Rada Mihalcea and Dragomir Radev

2:00–5:30 **T4: Automatic Spoken Document Processing for Retrieval and Browsing**
Ciprian Chelba and T. J. Hazen

2:00–5:30 **T5: Tutorial on Inductive Semi-supervised Learning Methods: with Applicability to Natural Language Processing**
Anoop Sarkar and Gholamreza Haffari

2:00–5:30 **T6: Automatic Semantic Role Labeling**
Scott Wen-tau Yih and Kristina Toutanova

6:30–9:30 Reception at NYU

Main Conference Program

Monday, June 5

9:00–9:10 Opening Session

9:10–10:10 **Keynote Speaker I: Joshua Goodman**
Email and Spam and Spim and Spat

10:10–10:40 Break

Machine Translation I

10:40–11:05 *Capitalizing Machine Translation*
Wei Wang, Kevin Knight and Daniel Marcu

11:05–11:30 *Do we need phrases? Challenging the conventional wisdom in Statistical Machine Translation*
Chris Quirk and Arul Menezes

11:30–11:55 *Improved Statistical Machine Translation Using Paraphrases*
Chris Callison-Burch, Philipp Koehn and Miles Osborne

11:55–12:20 *Segment Choice Models: Feature-Rich Models for Global Distortion in Statistical Machine Translation*
Roland Kuhn, Denis Yuen, Michel Simard, Patrick Paul, George Foster, Eric Joanis and Howard Johnson

Inference and Entailment

10:40–11:05 *Effectively Using Syntax for Recognizing False Entailment*
Rion Snow, Lucy Vanderwende and Arul Menezes

11:05–11:30 *Learning to recognize features of valid textual entailments*
Bill MacCartney, Trond Grenager, Marie-Catherine de Marneffe, Daniel Cer and Christopher D. Manning

11:30–11:55 *Acquisition of Verb Entailment from Text*
Viktor Pekar

11:55–12:20 *Acquiring Inference Rules with Temporal Constraints by Using Japanese Coordinated Sentences and Noun-Verb Co-occurrences*
Kentaro Torisawa

Monday, June 5 (continued)

Named Entity Recognition

10:40–11:05 *Role of Local Context in Automatic Deidentification of Ungrammatical, Fragmented Text*
Tawanda Sibanda, Ozlem Uzuner and Ozlem Uzuner

11:05–11:30 *Exploiting Domain Structure for Named Entity Recognition*
Jing Jiang and ChengXiang Zhai

11:30–11:55 *Named Entity Transliteration and Discovery from Multilingual Comparable Corpora*
Alexandre Klementiev and Dan Roth

11:55–12:20 *Reducing Weight Undertraining in Structured Discriminative Learning*
Charles Sutton, Michael Sindelar and Andrew McCallum

12:20–1:50 Lunch

Short Papers: Machine Translation, Multi-Lingual Speech

1:50–2:05 *Spectral Clustering for Example Based Machine Translation*
Rashmi Gangadharaiah, Ralf Brown and Jaime Carbonell

2:05–2:20 *Bridging the Inflection Morphology Gap for Arabic Statistical Machine Translation*
Andreas Zollmann, Venugopal Ashish and Vogel Stephan

2:20–2:35 *Arabic Preprocessing Schemes for Statistical Machine Translation*
Nizar Habash and Fatiha Sadat

2:35–2:50 *Thai Grapheme-Based Speech Recognition*
Paisarn Charoenpornsawat, Sanjika Hewavitharana and Tanja Schultz

2:50–3:05 *Story Segmentation of Broadcast News in English, Mandarin and Arabic*
Andrew Rosenberg and Julia Hirschberg

3:05–3:20 *Word Pronunciation Disambiguation using the Web*
Eiichiro Sumita and Fumiaki Sugaya

Monday, June 5 (continued)

Short Papers: Discourse/Dialogue

1:50–2:05 *Agreement/Disagreement Classification: Exploiting Unlabeled Data using Contrast Classifiers*
Sangyun Hahn, Richard Ladner and Mari Ostendorf

2:05–2:20 *Using Phrasal Patterns to Identify Discourse Relations*
Manami Saito, Kazuhide Yamamoto and Satoshi Sekine

2:20–2:35 *Evaluating Centering for Sentence Ordering in Two New Domains*
Nikiforos Karamanis

2:35–2:50 *Computational Modelling of Structural Priming in Dialogue*
David Reitter, Frank Keller and Johanna D. Moore

2:50–3:05 *Museli: A Multi-Source Evidence Integration Approach to Topic Segmentation of Spontaneous Dialogue*
Jaime Arguello and Carolyn Rose

3:05–3:20 *Automatic Recognition of Personality in Conversation*
Franois Mairesse and Marilyn Walker

Short Papers: Retrieval, Language Models

1:50–2:05 *Using the Web to Disambiguate Acronyms*
Eiichiro Sumita and Fumiaki Sugaya

2:05–2:20 *Lycos Retriever: An Information Fusion Engine*
Brian Ulicny

2:20–2:35 *BioEx: A Novel User-Interface that Accesses Images from Abstract Sentences*
Hong Yu and Minsuk Lee

2:35–2:50 *Selecting relevant text subsets from web-data for building topic specific language models*
Abhinav Sethy, Panayiotis Georgiou and Shrikanth Narayanan

2:50–3:05 *Factored Neural Language Models*
Andrei Alexandrescu and Katrin Kirchhoff

3:05–3:20 *Quantitative Methods for Classifying Writing Systems*
Gerald Penn and Travis Choma

3:20–3:50 Break

Monday, June 5 (continued)

Word Alignment

3:50–4:15 *A Maximum Entropy Approach to Combining Word Alignments*
Necip Fazil Ayan and Bonnie J. Dorr

4:15–4:40 *Alignment by Agreement*
Percy Liang, Ben Taskar and Dan Klein

4:40–5:05 *Word Alignment via Quadratic Assignment*
Simon Lacoste-Julien, Ben Taskar, Dan Klein and Michael I. Jordan

Semantics I

3:50–4:15 *An Empirical Study of the Behavior of Active Learning for Word Sense Disambiguation*
Jinying Chen, Andrew Schein, Lyle Ungar and Martha Palmer

4:15–4:40 *Unknown word sense detection as outlier detection*
Katrin Erk

4:40–5:05 *Understanding Temporal Expressions in Emails*
Benjamin Han, Donna Gates and Lori Levin

Parsing I

3:50–4:15 *Partial Training for a Lexicalized-Grammar Parser*
Stephen Clark and James Curran

4:15–4:40 *Effective Self-Training for Parsing*
David McClosky, Eugene Charniak and Mark Johnson

4:40–5:05 *Multilingual Dependency Parsing using Bayes Point Machines*
Simon Corston-Oliver, Anthony Aue, Kevin Duh and Eric Ringger

Tuesday, June 6

Parsing II

9:00–9:25 *Multilevel Coarse-to-Fine PCFG Parsing*
Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R. Shrivaths, Jeremy Moore, Michael Pozar and Theresa Vu

9:25–9:50 *A Fully-Lexicalized Probabilistic Model for Japanese Syntactic and Case Structure Analysis*
Daisuke Kawahara and Sadao Kurohashi

9:50–10:15 *Fully Parsing the Penn Treebank*
Ryan Gabbard, Seth Kulick and Mitchell Marcus

Discourse

9:00–9:25 *Exploiting Semantic Role Labeling, WordNet and Wikipedia for Coreference Resolution*
Simone Paolo Ponzetto and Michael Strube

9:25–9:50 *Identifying and Analyzing Judgment Opinions*
Soo-Min Kim and Eduard Hovy

9:50–10:15 *Learning to Detect Conversation Focus of Threaded Discussions*
Donghui Feng, Erin Shaw, Jihie Kim and Eduard Hovy

Spoken and Acoustic Aspects of Language

9:00–9:25 *Towards Automatic Scoring of Non-Native Spontaneous Speech*
Klaus Zechner and Isaac Bejar

9:25–9:50 *Unsupervised and Semi-supervised Learning of Tone and Pitch Accent*
Gina-Anne Levow

9:50–10:15 *Learning Pronunciation Dictionaries: Language Complexity and Word Selection Strategies*
John Kominek and Alan W Black

10:15–10:45 Break

Tuesday, June 6 (continued)

Machine Translation II

10:45–11:10 *Relabeling Syntax Trees to Improve Syntax-Based Machine Translation Quality*
Bryant Huang and Kevin Knight

11:10–11:35 *Grammatical Machine Translation*
Stefan Riezler and John T. Maxwell III

11:35–12:00 *Synchronous Binarization for Machine Translation*
Hao Zhang, Liang Huang, Daniel Gildea and Kevin Knight

Dialogue

10:45–11:10 *Modelling User Satisfaction and Student Learning in a Spoken Dialogue Tutoring System with Generic, Tutoring, and User Affect Parameters*
Kate Forbes-Riley and Diane Litman

11:10–11:35 *Comparing the Utility of State Features in Spoken Dialogue Using Reinforcement Learning*
Joel Tetreault and Diane Litman

11:35–12:00 *Backoff Model Training using Partially Observed Data: Application to Dialog Act Tagging*
Gang Ji and Jeff Bilmes

Relation Extraction

10:45–11:10 *Exploring Syntactic Features for Relation Extraction using a Convolution Tree Kernel*
Min Zhang, Jie Zhang and Jian Su

11:10–11:35 *Integrating Probabilistic Extraction Models and Data Mining to Discover Relations and Patterns in Text*
Aron Culotta, Andrew McCallum and Jonathan Betz

11:35–12:00 *Preemptive Information Extraction using Unrestricted Relation Discovery*
Yusuke Shinyama and Satoshi Sekine

12:00–1:30 Lunch

Tuesday, June 6 (continued)

Best Paper And Plenary Demo Presentations

1:30–2:00 *Probabilistic Context-Free Grammar Induction Based on Structural Zeros*
Mehryar Mohri and Brian Roark

2:00–2:30 *Prototype-Driven Learning for Sequence Models*
Aria Haghighi and Dan Klein

2:30–3:00 Plenary demos:

InfoMagnets: Making Sense of Corpus Data
Jamie Arguello and Carolyn Rose

Question Answering with Web, Mobile and Speech Interfaces
Edward Whittaker, Joanna Mrozinski, and Sadaoki Furui

From Pipedreams to Products and Promise!
Janet Baker and Patri Pugliese

3:00–3:15 Break

3:15–5:15 Posters and Demos

7:00 Banquet

Wednesday, June 7

9:00–10:00 **Keynote Speaker II: Diane Litman**
Spoken Dialogue for Intelligent Tutoring Systems: Opportunities and Challenges

10:00–10:30 Break

Morphology/Grammar Induction

10:30–10:55 *Learning Morphological Disambiguation Rules for Turkish*
Deniz Yuret and Ferhan Ture

10:55–11:20 *Cross-Entropy and Estimation of Probabilistic Context-Free Grammars*
Anna Corazza and Giorgio Satta

11:20–11:45 *Estimation of Consistent Probabilistic Context-free Grammars*
Mark-Jan Nederhof and Giorgio Satta

11:45–12:10 *A Better N-Best List: Practical Determinization of Weighted Finite Tree Automata*
Jonathan May and Kevin Knight

Wednesday, June 7 (continued)

Generation/Summarization/Question Answering

10:30–10:55 *Aggregation via Set Partitioning for Natural Language Generation*
Regina Barzilay and Mirella Lapata

10:55–11:20 *Incorporating Speaker and Discourse Features into Speech Summarization*
Gabriel Murray, Steve Renals, Jean Carletta and Johanna Moore

11:20–11:45 *Nuggeteer: Automatic Nugget-Based Evaluation using Descriptions and Judgements*
Gregory Marton and Alexey Radul

11:45–12:10 *Will Pyramids Built of Nuggets Topple Over?*
Jimmy Lin and Dina Demner-Fushman

Information Retrieval

10:30–10:55 *Creating a Test Collection for Citation-based IR Experiments*
Anna Ritchie, Simone Teufel and Stephen Robertson

10:55–11:20 *A Machine Learning based Approach to Evaluating Retrieval Systems*
Huyen-Trang Vu and Patrick Gallinari

11:20–11:45 *Language Model Information Retrieval with Document Expansion*
Tao Tao, Xuanhui Wang, Qiaozhu Mei and ChengXiang Zhai

11:45–12:10 *Towards Spoken-Document Retrieval for the Internet: Lattice Indexing For Large-Scale Web-Search Architectures*
Zheng-Yu Zhou, Peng Yu, Ciprian Chelba and Frank Seide

12:10–1:40 Lunch

1:40–2:30 NAACL Business Meeting

Wednesday, June 7 (continued)

Short Papers: Morphology/Syntax

2:30–2:45 *Subword-based Tagging by Conditional Random Fields for Chinese Word Segmentation*
Ruiqiang Zhang, Kikui Genichiro and sumita eiichiro

2:45–3:00 *Accurate Parsing of the Proposition Bank*
Gabriele Musillo and Paola Merlo

3:00–3:15 *Early Deletion of Fillers In Processing Conversational Speech*
Matthew Lease and Mark Johnson

3:15–3:30 *Parser Combination by Reparsing*
Kenji Sagae and Alon Lavie

Short Papers: Semantics

2:30–2:45 *Unsupervised Induction of Modern Standard Arabic Verb Classes*
Neal Snider and Mona Diab

2:45–3:00 *Word Domain Disambiguation via Word Sense Disambiguation*
Antonio Sanfilippo, Stephen Tratz and Michelle Gregory

3:00–3:15 *Evaluation of Utility of LSA for Word Sense Discrimination*
Esther Levin, Mehrbod Sharifi and Jerry Ball

3:15–3:30 *Semi-supervised Relation Extraction with Label Propagation*
Jinxiu Chen, Donghong Ji, Chew Lim Tan and Zhengyu Niu

Short Papers: Speech and Video Processing

2:30–2:45 *Initial Study on Automatic Identification of Speaker Role in Broadcast News Speech*
Yang Liu

2:45–3:00 *Extracting Salient Keywords from Instructional Videos Using Joint Text, Audio and Visual Cues*
Youngja Park and Ying Li

3:00–3:15 *Class Model Adaptation for Speech Summarisation*
Pierre Chatain, Edward Whittaker, Joanna Mrozinski and Sadaoki Furui

3:15–3:30 *Summarizing Speech Without Text Using Hidden Markov Models*
Sameer Maskey and Julia Hirschberg

3:30–4:00 Break

Wednesday, June 7 (continued)

Semantics II

4:00–4:25 *A fast finite-state relaxation method for enforcing global constraints on sequence decoding*
Roy Tromble and Jason Eisner

4:25–4:50 *Semantic role labeling of nominalized predicates in Chinese*
Nianwen Xue

4:50–5:15 *Learning for Semantic Parsing with Statistical Machine Translation*
Yuk Wah Wong and Raymond Mooney

Evaluation

4:00–4:25 *ParaEval: Using Paraphrases to Evaluate Summaries Automatically*
Liang Zhou, Chin-Yew Lin, Dragos Stefan Munteanu and Eduard Hovy

4:25–4:50 *Paraphrasing for Automatic Evaluation*
David Kauchak and Regina Barzilay

4:50–5:15 *An Information-Theoretic Approach to Automatic Evaluation of Summaries*
Chin-Yew Lin, Guihong Cao, Jianfeng Gao and Jian-Yun Nie

Processing in/for Language Models

4:00–4:25 *Cross Linguistic Name Matching in English and Arabic*
Andrew Freeman, Sherri Condon and Christopher Ackerman

4:25–4:50 *Language Model-Based Document Clustering Using Random Walks*
Gunes Erkan

4:50–5:15 *Unlimited vocabulary speech recognition for agglutinative languages*
Mikko Kurimo, Antti Puurula, Ebru Arisoy, Vesa Siivola, Teemu Hirsimki, Janne Pylkknen, Tanel Alume and Murat Saraclar

Workshops

Thursday, June 8

9:00–5:30 **WS01: The Tenth Conference on Computational Natural Language Learning (CoNLL-X)**, Day 1

9:00–5:30 **WS02: Document Understanding Conference (DUC)**, Day 1

9:00–5:30 **WS03: Interactive Question Answering**, Day 1

9:00–5:30 **WS04: Statistical Machine Translation**, Day 1

9:00–5:30 **WS05: Special Interest Group on Computational Phonology (SIGPHON)**

9:00–5:30 **WS06: BioNLP'06: Linking Natural Language Processing and Biology: Towards deeper biological literature analysis**

9:00–5:30 **WS08: Analyzing Conversations in Text and Speech (ACTS)**

9:00–5:30 **WS09: Third International Workshop on Scalable Natural Language Understanding (ScaNaLU 2006)**

Friday, June 9

9:00–5:30 **WS01: The Tenth Conference on Computational Natural Language Learning (CoNLL-X)**, Day 2

9:00–5:30 **WS02: Document Understanding Conference (DUC)**, Day 2

9:00–5:30 **WS03: Interactive Question Answering**, Day 2

9:00–5:30 **WS04: Statistical Machine Translation**, Day 2

9:00–5:30 **WS10: Computationally Hard Problems and Joint Inference in Speech and Language Processing**

9:00–5:30 **WS11: First International Workshop on Medical Speech Translation**

9:00–5:30 **WS12: Textgraphs: Graph-based Algorithms for Natural Language Processing**

Keynote Speaker:

Joshua Goodman
Microsoft Research

Speaking on:

Email and Spam and Spim and Spat

Abstract

Email is the number one activity that people do on the internet: 74% of internet users check their email on an average day. Email use in offices has more than doubled since 2000, and is now over 8 hours a week. There are many great NLP problems for email, like automatic clustering and foldering, search, prioritization, automatically finding keywords within messages, finding addresses, and summarization. Spam is the number one problem for email. I?ll talk about how spam filters work, and the current open problems, as well as other kinds of abuse like chat spam (Spat), IM spam (Spim), blog comment spam (Blat), etc. all of which make great NLP problems.

Email and abuse problems like spam can be some of the most exciting for research: they inspire us to work on new problems we would otherwise not have found. We are exploring areas like adversarial learning, learning with unbalanced costs, and learning with partial user feedback. Shipping solutions to these problems is both surprisingly hard and surprisingly fun. For NLP Researchers, the hardest constraint is that products ship in about 20 languages. By carefully choosing tools like word clustering that are easy to build in many languages, instead of similar tools like taggers that may not exist everywhere, we increase the chance of shipping. When we have actually built complete systems and given them to users, we have found several new and interesting problems in the most exciting way, by shipping solutions that don?t work the first time around.

Bio

Joshua Goodman is a Principal Researcher in the Machine Learning and Applied Statistic group at Microsoft Research, where he runs a team focused on Learning for Messaging and Adversarial Problems. Spam filters he helped develop stop over a billion spam messages per day. He has also worked on language modeling and machine learning, and has a Ph.D. in Computer Science from Harvard University for his work on Statistical Parsing. He helped start and is now President of the Conference on Email and Anti-Spam.

Keynote Speaker:

Diane Litman
University of Pittsburgh

Speaking on:

Spoken Dialogue for Intelligent Tutoring Systems: Opportunities and Challenges

Abstract

In recent years, the development of intelligent tutoring dialogue systems has become more prevalent, in an attempt to close the performance gap between human and computer tutors. With advances in speech technology, several systems have begun to incorporate spoken language capabilities, hypothesizing that adding speech technology will promote student learning by enhancing communication richness. Tutoring applications differ in many ways, however, from the types of applications for which spoken dialogue systems are typically developed. This talk will illustrate some of the opportunities and challenges in this area, focusing on issues such as affective reasoning, discourse analysis, error handling, and performance evaluation.

Bio

Diane Litman is Professor of Computer Science, as well as Research Scientist with the Learning Research and Development Center, at the University of Pittsburgh. Previously, Dr. Litman was a member of the Artificial Intelligence Principles Research Department, AT&T Labs - Research (formerly Bell Laboratories); she was also an Assistant Professor of Computer Science at Columbia University. Dr. Litman received her Ph.D. degree in Computer Science from the University of Rochester. Her current research focuses on enhancing the effectiveness of tutorial dialogue systems through the use of spoken language processing, affective computing, and machine learning. She has collaborated on the development of spoken dialogue systems in multiple application areas, including intelligent tutoring (ITSPOKE), chat (CobotDS) and database/web access (NJFun and TOOT). Dr. Litman has been Chair of the North American Chapter of the Association for Computational Linguistics, a member of the Executive Committee of the Association for Computational Linguistics, and a member of the editorial boards of Computational Linguistics and User Modeling and User-Adapted Interaction.

Capitalizing Machine Translation

Wei Wang and **Kevin Knight** and **Daniel Marcu**
Language Weaver, Inc.
4640 Admiralty Way, Suite 1210
Marina del Rey, CA, 90292
{wwang, kknight, dmarcu}@languageweaver.com

Abstract

We present a probabilistic bilingual capitalization model for capitalizing machine translation outputs using conditional random fields. Experiments carried out on three language pairs and a variety of experiment conditions show that our model significantly outperforms a strong monolingual capitalization model baseline, especially when working with small datasets and/or European language pairs.

1 Introduction

Capitalization is the process of recovering case information for texts in lowercase. It is also called truecasing (Lita et al., 2003). Usually, capitalization itself tries to improve the legibility of texts. It, however, can affect the word choice or order when interacting with other models. In natural language processing, a good capitalization model has been shown useful for tasks like name entity recognition, automatic content extraction, speech recognition, modern word processors, and machine translation (MT).

Capitalization can be viewed as a sequence labeling process. The input to this process is a sentence in lowercase. For each lowercased word in the input sentence, we have several available capitalization tags: initial capital (IU), all uppercase (AU), all lowercase (AL), mixed case (MX), and all having no case (AN). The output of capitalization is a capitalization tag sequence. Associating a tag in the output with the corresponding lowercased word in the input results in a surface form of the word. For example, we can tag the input sentence "click ok to save your changes to /home/doc." into "click_IU ok_AU to_AL save_AL your_AL changes_AL to_AL /home/doc_MX ._AN", getting the surface form "Click OK to save your changes to /home/DOC .".

A capitalizer is a tagger that recovers the capitalization tag for each input lowercased word, outputting a well-capitalized sentence. Since each lowercased word can have more than one tag, and associating a tag with a lowercased word can result in more than one surface form (e.g., /home/doc_MX can be either /home/DOC or /home/Doc), we need a capitalization model to solve the capitalization ambiguities. For example, Lita et al. (2003) use a trigram language model estimated from a corpus with case information; Chelba and Acero (2004) use a maximum entropy Markov model (MEMM) combining features involving words and their cases.

Capitalization models presented in most previous approaches are monolingual because the models are estimated only from monolingual texts. However, for capitalizing machine translation outputs, using only monolingual capitalization models is not enough. For example, if the sentence "click ok to save your changes to /home/doc ." in the above example is the translation of the French sentence "CLIQUEZ SUR OK POUR ENREGISTRER VOS MODIFICATIONS DANS /HOME/DOC .", the correct capitalization result should probably be "CLICK OK TO SAVE YOUR CHANGES TO /HOME/DOC .", where all words are in all upper-case. Without looking into the case

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 1–8,
New York, June 2006.

of the MT input, we can hardly get the correct capitalization result.

Although monolingual capitalization models in previous work can apply to MT output, a bilingual model is more desirable. This is because MT outputs usually strongly preserve case from the input, and because monolingual capitalization models do not always perform as well on badly translated text as on well-formed syntactic texts.

In this paper, we present a bilingual capitalization model for capitalizing machine translation outputs using conditional random fields (CRFs) (Lafferty et al., 2001). This model exploits case information from both the input sentence (source) and the output sentence (target) of the MT system. We define a series of feature functions to incorporate capitalization knowledge into the model.

Experimental results are shown in terms of BLEU scores of a phrase-based SMT system with the capitalization model incorporated, and in terms of capitalization precision. Experiments are performed on both French and English targeted MT systems with large-scale training data. Our experimental results show that the CRF-based bilingual capitalization model performs better than a strong baseline capitalizer that uses a trigram language model.

2 Related Work

A simple capitalizer is the 1-gram tagger: the case of a word is always the most frequent one observed in training data, with the exception that the sentence-initial word is always capitalized. A 1-gram capitalizer is usually used as a baseline for capitalization experiments (Lita et al., 2003; Kim and Woodland, 2004; Chelba and Acero, 2004).

Lita et al. (2003) view capitalization as a lexical ambiguity resolution problem, where the lexical choices for each lowercased word happen to be its different surface forms. For a lowercased sentence e, a trigram language model is used to find the best capitalization tag sequence T that maximizes $p(T, e) = p(E)$, resulting in a case-sensitive sentence E. Besides local trigrams, sentence-level contexts like sentence-initial position are employed as well.

Chelba and Acero (2004) frame capitalization as a sequence labeling problem, where, for each lowercased sentence e, they find the label sequence T that maximizes $p(T|e)$. They use a maximum entropy Markov model (MEMM) to combine features of words, cases and context (i.e., tag transitions).

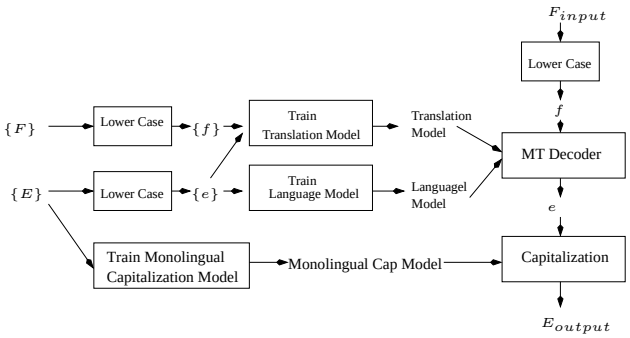


Figure 1: The monolingual capitalization scheme employed by most statistical MT systems.

Gale et al. (1994) report good results on capitalizing 100 words. Mikheev (1999) performs capitalization using simple positional heuristics.

3 Monolingual Capitalization Scheme

Translation and capitalization are usually performed in two successive steps because removing case information from the training of translation models substantially reduces both the source and target vocabulary sizes. Smaller vocabularies lead to a smaller translation model with fewer parameters to learn. For example, if we do not remove the case information, we will have to deal with at least nine probabilities for the English-French word pair (click, cliquez). This is because either "click" or "cliquez" can have at least three tags (IU, AL, AU), and thus three surface forms. A smaller translation model requires less training data, and can be estimated more accurately than otherwise from the same amount of training data. A smaller translation model also means less memory usage.

Most statistical MT systems employ the monolingual capitalization scheme as shown in Figure 1. In this scheme, the translation model and the target language model are trained from the lowercased corpora. The capitalization model is trained from the case-sensitive target corpus. In decoding, we first turn input into lowercase, then use the decoder to generate the lowercased translation, and finally ap-

HYDRAULIC HEADER TILT CYLINDER KIT Kit de vérin d'inclinaison hydraulique de la plate-forme
haut-parleur avant droit + HAUT-PARLEUR AVANT DROIT +
Seat Controls, Standard COMMANDES DU SIGE, STANDARD
loading a saved legend Chargement d'une légende sauvegarde

Table 1: Errors made by monolingual capitalization model. Each row contains a pair of MT input and MT output.

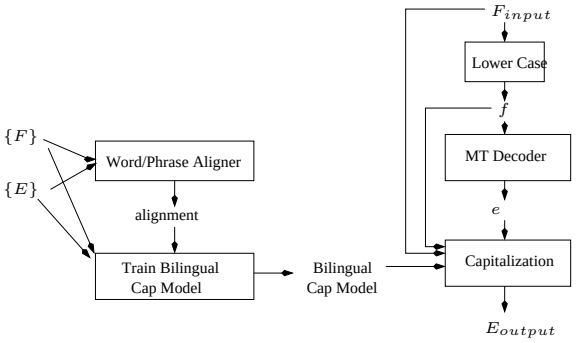


Figure 2: A bilingual capitalization scheme.

ply the capitalization model to recover the case of the decoding output.

The monolingual capitalization scheme makes many errors as shown in Table 1. Each cell in the table contains the MT-input and the MT-output. These errors are due to the capitalizer does not have access to the source sentence.

Regardless, estimating mixed-cased translation models, however, is a very interesting topic and worth future study.

4 Bilingual Capitalization Model

4.1 The Model

Our probabilistic bilingual capitalization model exploits case information from both the input sentence to the MT system and the output sentence from the system (see Figure 2). An MT system translates a *capitalized* sentence F into a lowercased sentence e. A statistical MT system can also provide the alignment A between the input F and the output e; for example, a statistical phrase-based MT system could provide the phrase boundaries in F and e, and also the alignment between the phrases.[1]

[1]We shall explain our capitalization model within the phrase-based SMT framework, the model, however, could be adapted to syntax-based machine translation, too. To this end, the translational correspondence is described within a translation rule, i.e., (Galley et al., 2004) (or a synchronous production), rather than a translational phrase pair; and the training data will be derivation forests, instead of the phrase-aligned bilingual corpus.

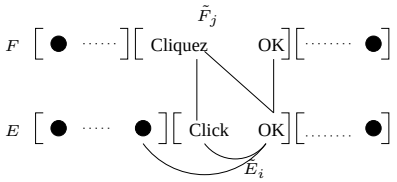


Figure 3: Alignment graph. Brackets mean phrase boundaries.

The bilingual capitalization algorithm recovers the capitalized sentence E from e, according to the input sentence F, and the alignment A. Formally, we look for the best capitalized sentence E^* such that

$$E^* = \arg\max_{E \in \mathbf{GEN}(e)} p(E|F, A) \quad (1)$$

where $\mathbf{GEN}(e)$ is a function returning the set of possible capitalized sentences consistent with e. Notice that e does not appear in $p(E|F, A)$ because we can uniquely obtain e from E. $p(E|F, A)$ is the capitalization model of concern in this paper.[2]

To further decompose the capitalization model $p(E|F, A)$, we make some assumptions. As shown in Figure 3, input sentence F, capitalized output E, and their alignment can be viewed as a graph. Vertices of the graph correspond to words in F and E. An edge connecting a word in F and a word in E corresponds to a word alignment. An edge between two words in E represents the dependency between them captured by monolingual n-gram language models. We also assume that both E and F have phrase boundaries available (denoted by the square brackets), and that A is the phrase alignment. In Figure 3, $\tilde{F}_j$ is the j-th phrase of F, $\tilde{E}_i$ is the i-th phrase of E, and they align to each other. We do not require a word alignment; instead we find it reasonable to think that a word in $\tilde{E}_i$ can be aligned to any

[2]The capitalization model $p(E|F, A)$ itself does not require the existence of e. This means that in principle this model can also be viewed as a capitalized translation model that performs translation and capitalization in an integrated step. In our paper, however, we consider the case where the machine translation output e is given, which is reflected by the the fact that **GEN**(e) takes e as input in Formula 1.

word in $\tilde{F}_j$. A probabilistic model defined on this graph is a Conditional Random Field. Therefore, it is natural to formulate the bilingual capitalization model using CRFs:[3]

$$p_{\overline{\lambda}}(E|F,A) = \frac{1}{Z(F,A,\overline{\lambda})} \exp\left(\sum_{i=1}^{I} \lambda_i f_i(E,F,A)\right) \quad (2)$$

where

$$Z(F,A,\overline{\lambda}) = \sum_{E \in \mathbf{GEN}(e)} \exp\left(\sum_{i=1}^{I} \lambda_i f_i(E,F,A)\right) \quad (3)$$

$f_i(E,F,A), i = 1...I$ are the I features, and $\overline{\lambda} = (\lambda_1, ..., \lambda_I)$ is the feature weight vector. Based on this capitalization model, the decoder in the capitalizer looks for the best E^* such that

$$E^* = \arg\max_{E \in \mathbf{GEN}(e,F)} \sum_{i=1}^{I} \lambda_i f_i(E,F,A) \quad (4)$$

4.2 Parameter Estimation

Following Roark et al. (2004), Lafferty et al. (2001) and Chen and Rosenfeld (1999), we are looking for the set of feature weights $\overline{\lambda}$ maximizing the regularized log-likelihood $LL_R(\overline{\lambda})$ of the training data $\{E^{(n)}, F^{(n)}, A^{(n)}, n = 1, ..., N\}$.

$$LL_R(\overline{\lambda}) = \sum_{n=1}^{N} \log p\left(E^{(n)}|F^{(n)}, A^{(n)}\right) - \frac{||\overline{\lambda}||^2}{2\sigma^2} \quad (5)$$

The second term at the right-hand side of Formula 5 is a zero-mean Gaussian prior on the parameters. σ is the variance of the Gaussian prior dictating the cost of feature weights moving away from the mean — a smaller value of σ keeps feature weights closer to the mean. σ can be determined by linear search on development data.[4] The use of the Gaussian prior term in the objective function has been found effective in avoiding overfitting, leading to consistently better results. The choice of LL_R as an objective function can be justified as maximum a-posteriori (MAP) training within a Bayesian approach (Roark et al., 2004).

[3]We chose CRFs over other sequence labeling models (i.e. MEMM) because CRFs have no label bias and we do not need to compute the partition function during decoding.

[4]In our experiment, we use an empirical value $\sigma = 0.5$ as in (Roark et al., 2004).

4.3 Feature Functions

We define features based on the alignment graph in Figure 3. Each feature function is defined on a word.

Monolingual language model feature. The monolingual LM feature of word E_i is the logarithm of the probability of the n-gram ending at E_i:

$$f_{\mathrm{LM}}(E_i,F,A) = \log p(E_i|E_{i-1}, ..., E_{i-n+1}) \quad (6)$$

p should be appropriately smoothed such that it never returns zero.

Capitalized translation model feature. Suppose E phrase "Click OK" is aligned to F phrase "Cliquez OK". The capitalized translation model feature of "Click" is computed as $\log p(\mathrm{Click}|\mathrm{Cliquez}) + \log p(\mathrm{Click}|\mathrm{OK})$. "Click" is assumed to be aligned to any word in the F phrase. The larger the probability that "Click" is translated from an F word, i.e., "Cliquez", the more chances that "Click" preserves the case of "Cliquez". Formally, for word E_i, and an aligned phrase pair $\tilde{E}_l$ and $\tilde{F}_m$, where $E_i \in \tilde{E}_l$, the capitalized translation model feature of E_i is

$$f_{\mathrm{cap\cdot t1}}(E_i,F,A) = \log \sum_{k=1}^{|\tilde{F}_m|} p(E_i|\tilde{F}_{m,k}) \quad (7)$$

$p(E_i|\tilde{F}_{m,k})$ is the capitalized translation table. It needs smoothing to avoid returning zero, and is estimated from a word-aligned bilingual corpus.

Capitalization tag translation feature. The feature value of E word "Click" aligning to F phrase "Cliquez OK" is $\log p(\mathrm{IU}|\mathrm{IU})p(\mathrm{click}|\mathrm{cliquez}) + \log p(\mathrm{IU}|\mathrm{AU})p(\mathrm{click}|\mathrm{ok})$. We see that this feature is less specific than the capitalized translation model feature. It is computed in terms of the tag translation probability and the lowercased word translation probability. The lowercased word translation probability, i.e., $p(\mathrm{click}|\mathrm{ok})$, is used to decide how much of the tag translation probability, i.e., $p(\mathrm{IU}|\mathrm{AU})$, will contribute to the final decision. The smaller the word translation probability, i.e., $p(\mathrm{click}|\mathrm{ok})$, is, the smaller the chance that the surface form of "click"

preserves case from that of "ok". Formally, this feature is defined as

$$f_{\text{cap·tag·t1}}(E_i, F, A) = \log \sum_{k=1}^{|\tilde{f}_m|} p(e_i|\tilde{f}_{m,k}) \times p(\tau(E_i)|\tau(\tilde{F}_{m,k})) \quad (8)$$

$p(e_i|\tilde{f}_{m,k})$ is the t-table over lowercased word pairs, which is the usual "t-table" in a SMT system. $p(\tau(E_i)|\tau(\tilde{F}_{m,k}))$ is the probability of a target capitalization tag given a source capitalization tag and can be easily estimated from a word-aligned bilingual corpus. This feature attempts to help when $f_{\text{cap-t1}}$ fails (i.e., the capitalized word pair is unseen). Smoothing is also applied to both $p(e_i|\tilde{f}_{m,k})$ and $p(\tau(E_i)|\tau(\tilde{F}_{m,k}))$ to handle unseen words (or word pairs).

Upper-case translation feature. Word E_i is in all upper case if all words in the corresponding F phrase $\tilde{F}_m$ are in upper case. Although this feature can also be captured by the capitalization tag translation feature in the case where an AU tag in the input sentence is most probably preserved in the output sentence, we still define it to emphasize its effect. This feature aims, for example, to translate "ABC XYZ" into "UUU VVV" even if all words are unseen.

Initial capitalization feature. An E word is initially capitalized if it is the first word *that contains* letters in the E sentence. For example, for sentence "• Please click the button" that starts with a bullet, the initial capitalization feature value of word "please" is 1 because "•" does not contain a letter.

Punctuation feature template. An E word is initially capitalized if it follows a punctuation mark. Non-sentence-ending punctuation marks like commas will usually get negative weights.

As one can see, our features are "coarse-grained" (e.g., the language model feature). In contrast, Kim and Woodland (2004) and Roark et al. (2004) use "fine-grained" features. They treat each n-gram as a feature for, respectively, monolingual capitalization and language modeling. Feature weights tuned at a fine granularity may lead to better accuracy, but they require much more training data, and result in much slower training speed, especially for large-scale learning problems. Coarse-grained features enable us to efficiently get the feature values from a very large training corpus, and quickly tune the weights on small development sets. For example, we can train a bilingual capitalization model on a 70 million-word corpus in several hours with the coarse-grained features presented above, but in several days with fine-grained n-gram count features.

4.4 The GEN Function

Function **GEN** generates the set of case-sensitive candidates from a lowercased token. For example $\mathbf{GEN}(\texttt{mt}) = \{\texttt{mt}, \texttt{mT}, \texttt{Mt}, \texttt{MT}\}$. The following heuristics can be used to reduce the range of **GEN**. The returned set of **GEN** on a lower-cased token w is the union of: (i) $\{w, AU(w), IU(w)\}$, (ii) $\{v|v \; is \; seen \; in \; training \; data \; and \; AL(v) = w\}$, and (iii) $\{\tilde{F}_{m,k}|AL(\tilde{F}_{m,k}) = AL(w)\}$. The heuristic (iii) is designed to provide more candidates for w when it is translated from a very strange input word $\tilde{F}_{m,k}$ in the F phrase $\tilde{F}_m$ that is aligned to the phrase that w is in. This heuristic creates good capitalization candidates for the translation of URLs, file names, and file paths.

5 Generating Phrase-Aligned Training Data

Training the bilingual capitalization model requires a bilingual corpus with phrase alignments, which are usually produced from a phrase aligner. In practice, the task of phrase alignment can be quite computationally expensive as it requires to translate the entire training corpus; also a phrase aligner is not always available. We therefore generate the training data using a naïve phrase aligner (NPA) instead of resorting to a real one.

The input to the NPA is a word-aligned bilingual corpus. The NPA stochastically chooses for each sentence pair one segmentation and phrase alignment that is consistent with the word alignment. An aligned phrase pair is consistent with the word alignment if neither phrase contains any word aligning to a word outside the other phrase (Och and Ney, 2004). The NPA chunks the source sentence into phrases according to a probabilistic distribution over source phrase lengths. This distribution can be obtained from the trace output of a phrase-based MT

Languages	Entire Corpus (#W) Training	Dev	Test-Prec.	Test-BLEU (#sents)
E→F (IT)	62M	13K	15K	763
F→E (news)	144M	11K	22K	241
C→E (news)	50M	8K	17K	919

Table 2: Corpora used in experiments.

decoder on a small development set. The NPA has to retry if the current source phrase cannot find any consistent target phrase. Unaligned target words are attached to the left phrase. Heuristics are employed to prevent the NPA from not coming to a solution. Obviously, the NPA is a special case of the phrase extractor in (Och and Ney, 2004) in that it considers only one phrase alignment rather than all possible ones.

Unlike a real phrase aligner, the NPA need not wait for the training of the translation model to finish, making it possible for parallelization of translation model training and capitalization model training. However, we believe that a real phrase aligner may make phrase alignment quality higher.

6 Experiments

6.1 Settings

We conducted capitalization experiments on three language pairs: English-to-French (E→F) with a bilingual corpus from the Information Technology (IT) domain; French-to-English (F→E) with a bilingual corpus from the general news domain; and Chinese-to-English (C→E) with a bilingual corpus from the general news domain as well. Each language pair comes with a training corpus, a development corpus and two test sets (see Table 2). Test-Precision is used to test the capitalization precision of the capitalizer on well-formed sentences drawn from genres similar to those used for training. Test-BLEU is used to assess the impact of our capitalizer on end-to-end translation performance; in this case, the capitalizer may operate on ungrammatical sentences. We chose to work with these three language pairs because we wanted to test our capitalization model on both English and French target MT systems and in cases where the source language has no case information (such as in Chinese).

We estimated the feature functions, such as the log probabilities in the language model, from the training set. Kneser-Ney smoothing (Kneser and Ney, 1995) was applied to features f_{LM}, $f_{\mathrm{cap{\cdot}t1}}$, and $f_{\mathrm{cap{\cdot}tag{\cdot}t1}}$. We trained the feature weights of the CRF-based bilingual capitalization model using the development set. Since estimation of the feature weights requires the phrase alignment information, we efficiently applied the NPA on the development set.

We employed two LM-based capitalizers as baselines for performance comparison: a unigram-based capitalizer and a strong trigram-based one. The unigram-based capitalizer is the usual baseline for capitalization experiments in previous work. The trigram-based baseline is similar to the one in (Lita et al., 2003) except that we used Kneser-Ney smoothing instead of a mixture.

A phrase-based SMT system (Marcu and Wong, 2002) was trained on the bitext. The capitalizer was incorporated into the MT system as a post-processing module — it capitalizes the lowercased MT output. The phrase boundaries and alignments needed by the capitalizer were automatically inferred as part of the decoding process.

6.2 BLEU and Precision

We measured the impact of our capitalization model in the context of an end-to-end MT system using BLEU (Papineni et al., 2001). In this context, the capitalizer operates on potentially ill-formed, MT-produced outputs.

To this end, we first integrated our bilingual capitalizer into the phrase-based SMT system as a post-processing module. The decoder of the MT system was modified to provide the capitalizer with the case-preserved source sentence, the lowercased translation, and the phrase boundaries and their alignments. Based on this information, our bilingual capitalizer recovers the case information of the lowercased translation, outputting a capitalized target sentence. The case-restored machine translations were evaluated against the target test-BLEU set. For comparison, BLEU scores were also computed for an MT system that used the two LM-based baselines.

We also assessed the performance of our capitalizer on the task of recovering case information for well-formed grammatical texts. To this end, we used the precision metric that counted the number of cor-

rectly capitalized words produced by our capitalizer on well-formed, lowercased input

$$precision = \frac{\#correctly\ capitalized\ words}{\#total\ words} \quad (9)$$

To obtain the capitalization precision, we implemented the capitalizer as a standalone program. The inputs to the capitalizer were triples of a case-preserved source sentence, a lowercased target sentence, and phrase alignments between them. The output was the case-restored version of the target sentence. In this evaluation scenario, the capitalizer output and the reference differ only in case information — word choices and word orders between them are the same. Testing was conducted on Test-Precision. We applied the NPA to the Test-Precision set to obtain the phrases and their alignments because they were needed to trigger the features in testing. We used a Test-Precision set that was different from the Test-BLEU set because word alignments were by-products only of training of translation models on the MT training data and we could not put the Test-BLEU set into the MT training data. Rather than implementing a standalone word aligner, we randomly divided the MT training data into three non-overlapping sets: Test-Precision set, CRF capitalizer training set and dev set.

6.3 Results

The performance comparisons between our CRF-based capitalizer and the two LM-based baselines are shown in Table 3 and Table 4. Table 3 shows the BLEU scores, and Table 4 shows the precision. The BLEU upper bounds indicate the ceilings that a perfect capitalizer can reach, and are computed by ignoring the case information in both the capitalizer outputs and the reference. Obviously, the precision upper bounds for all language pairs are 100%.

The precision and end-to-end BLEU based comparisons show that, for European language pairs, the CRF-based bilingual capitalization model outperforms significantly the strong LM-based baseline. We got more than one BLEU point improvement on the MT translation between English and French, a 34% relative reduction in capitalization error rate for the French-to-English language pair, and a 42% relative error rate reduction for the English-to-French language pair. These results show that source language information provides significant help for capitalizing machine translation outputs. The results also show that when the source language does not have case, as in Chinese, the bilingual model equals a monolingual one.

The BLEU difference between the CRF-based capitalizer and the trigram one were larger than the precision difference. This indicates that the CRF-based capitalizer performs much better on non-grammatical texts that are generated from an MT system due to the bilingual feature of the CRF capitalizer.

6.4 Effect of Training Corpus Size

The experiments above were carried out on large data sets. We also conducted experiments to examine the effect of the training corpus size on capitalization precision. Figure 4 shows the effects. The experiment was performed on the E→F corpus. The bilingual capitalizer performed significantly better when the training corpus size was small (e.g., under 8 million words). This is common in many domains: when the training corpus size increases, the difference between the two capitalizers decreases.

7 Conclusions

In this paper, we have studied how to exploit bilingual information to improve capitalization performance on machine translation output, and evaluated the improvement over traditional methods that use only monolingual language models.

We first presented a probabilistic bilingual capitalization model for capitalizing machine translation outputs using conditional random fields. This model exploits bilingual capitalization knowledge as well as monolingual information. We defined a series of feature functions to incorporate capitalization knowledge into the model.

We then evaluated our CRF-based bilingual capitalization model both on well-formed texts in terms of capitalization precision, and on possibly ungrammatical end-to-end machine translation outputs in terms of BLEU scores. Experiments were performed on both French and English target MT systems with large-scale training data. Our experimental results showed that the CRF-based bilingual cap-

Translation	BLEU Scores			
	Unigram Capitalizer	Trigram Capitalizer	CRF-based Capitalizer	Upper Bound
F→E	24.96	26.73	**27.92**	28.85
E→F	32.63	34.66	**36.10**	36.17
C→E	23.81	25.92	**25.89**	-

Table 3: Impact of CRF-based capitalizer on end-to-end translation performance compared with two LM-based baselines.

Translation	Capitalization Precision (%)		
	Unigram capitalizer	Trigram capitalizer	CRF-based capitalizer
F→E	94.03	98.79	**99.20**
E→F	91.52	98.47	**99.11**
C→E	90.77	96.40	**96.76**

Table 4: Impact of CRF-based capitalizer on capitalization precision compared with two LM-based baselines.

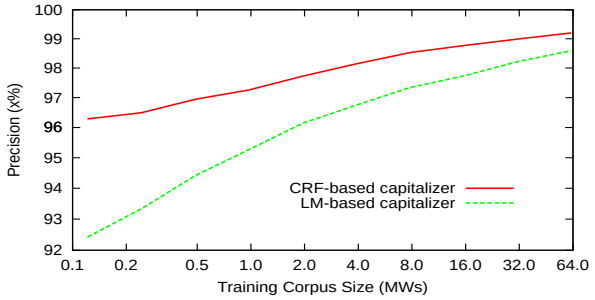


Figure 4: Capitalization precision with respect to size of training corpus. LM-based capitalizer refers to the trigram-based one. Results were on E→F corpus.

italization model performs significantly better than a strong baseline, monolingual capitalizer that uses a trigram language model.

In all experiments carried out at Language Weaver with customer (or domain specific) data, MT systems trained on lowercased data coupled with the CRF bilingual capitalizer described in this paper consistently outperformed both MT systems trained on lowercased data coupled with a strong monolingual capitalizer and MT systems trained on mixed-cased data.

References

Ciprian Chelba and Alex Acero. 2004. Adaptation of maximum entroy capitalizer: Little data can help a lot. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Barcelona, Spain.

Stanley Chen and Ronald Rosenfeld. 1999. A Gaussian prior for smoothing Maximum Entropy models. Technical Report CMUCS-99-108, Carnegie Mellon University.

William A. Gale, Kenneth W. Church, and David Yarowsky. 1994. Discrimination decisions for 100,000-dimensional spaces. In *Current issues in computational linguistics*.

M. Galley, M. Hopkins, K. Knight, and D. Marcu. 2004. What's in a Translation Rule? In *Proceedings of the Human Language Technology Conference and the North American Association for Computational Linguistics (HLT-NAACL)*, Boston, Massachusetts.

Ji-Hwan Kim and Philip C. Woodland. 2004. Automatic capitalization generation for speech input. *Computer Speech and Language*, 18(1):67–90, January.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 1995*, pages 181–184, Detroit, Michigan. IEEE.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fi elds: Probabilistic models for segmentation and labeling sequence data.

Lucian Vlad Lita, Abe Ittycheriah, Salim Roukos, and Nanda Kambhatla. 2003. tRuEcasIng. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, Sapporo, Japan, July.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Philadelphia, PA.

A. Mikheev. 1999. A knowledge-free method fro capitalized word disambiguation. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, College Park, Maryland, June.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: A method for automatic evaluation of Machine Translation. Technical Report RC22176, IBM, September.

Brian Roark, Murat Saraclar, Michael Collins, and Mark Johnson. 2004. Discriminative language modeling with conditional random fi eld and the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, Barcelona, Spain.

Do we need phrases? Challenging the conventional wisdom in Statistical Machine Translation

Chris Quirk and **Arul Menezes**
Microsoft Research
One Microsoft Way
Redmond, WA 98052 USA
{chrisq,arulm}@microsoft.com

Abstract

We begin by exploring theoretical and practical issues with phrasal SMT, several of which are addressed by syntax-based SMT. Next, to address problems not handled by syntax, we propose the concept of a Minimal Translation Unit (MTU) and develop MTU sequence models. Finally we incorporate these models into a syntax-based SMT system and demonstrate that it improves on the state of the art translation quality within a theoretically more desirable framework.

1. Introduction

The last several years have seen phrasal statistical machine translation (SMT) systems outperform word-based approaches by a wide margin (Koehn 2003). Unfortunately the use of phrases in SMT is beset by a number of difficult theoretical and practical problems, which we attempt to characterize below. Recent research into syntax-based SMT (Quirk and Menezes 2005; Chiang 2005) has produced promising results in addressing some of the problems; research motivated by other statistical models has helped to address others (Banchs et al. 2005). We refine and unify two threads of research in an attempt to address all of these problems simultaneously. Such an approach proves both theoretically more desirable and empirically superior.

In brief, Phrasal SMT systems employ phrase pairs automatically extracted from parallel corpora. To translate, a source sentence is first partitioned into a sequence of phrases $I = \boldsymbol{s}_1 \ldots \boldsymbol{s}_I$. Each source phrase $\boldsymbol{s}_i$ is then translated into a target phrase $\boldsymbol{t}_i$. Finally the target phrases are permuted, and the translation is read off in order. Beam search is used to approximate the optimal translation. We refer the reader to Keohn et al. (2003) for a detailed description. Unless otherwise noted, the following discussion is generally applicable to Alignment Template systems (Och and Ney 2004) as well.

1.1. Advantages of phrasal SMT

Non-compositionality
Phrases capture the translations of idiomatic and other non-compositional fixed phrases as a unit, side-stepping the need to awkwardly reconstruct them word by word. While many words can be translated into a single target word, common everyday phrases such as the English *password* translating as the French *mot de passe* cannot be easily subdivided. Allowing such translations to be first class entities simplifies translation implementation and improves translation quality.

Local re-ordering
Phrases provide memorized re-ordering decisions. As previously noted, translation can be conceptually divided into two steps: first, finding a set of phrase pairs that simultaneously covers the source side and provides a bag of translated target phrases; and second, picking an order for those target phrases. Since phrase pairs consist of memorized substrings of the training data, they are very likely to produce correct local re-orderings.

Contextual information
Many phrasal translations may be easily subdivided into word-for-word translation, for instance the English phrase *the cabbage* may be translated word-for-word as *le chou*. However we note that *la* is also a perfectly reasonable word-for-word translation of *the*, yet *la chou* is not a grammatical French string. Even when a phrase appears compositional, the incorporation of contextual information often improves translation

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 9–16,
New York, June 2006.

quality. Phrases are a straightforward means of capturing local context.

1.2. Theoretical problems with phrasal SMT

Exact substring match; no discontiguity
Large fixed phrase pairs are effective when an exact match can be found, but are useless otherwise. The alignment template approach (where phrases are modeled in terms of word classes instead of specific words) provides a solution at the expense of truly fixed phrases. Neither phrasal SMT nor alignment templates allow discontiguous translation pairs.

Global re-ordering
Phrases do capture local reordering, but provide no global re-ordering strategy, and the number of possible orderings to be considered is not lessened significantly. Given a sentence of *n* words, if the average target phrase length is 4 words (which is unusually high), then the re-ordering space is reduced from *n*! to only (*n*/4)!: still impractical for exact search in most sentences. Systems must therefore impose some limits on phrasal reordering, often hard limits based on distance as in Koehn et al. (2003) or some linguistically motivated constraint, such as ITG (Zens and Ney, 2004). Since these phrases are not bound by or even related to syntactic constituents, linguistic generalizations (such as SVO becoming SOV, or prepositions becoming postpositions) are not easily incorporated into the movement models.

Probability estimation
To estimate the translation probability of a phrase pair, several approaches are used, often concurrently as features in a log-linear model. Conditional probabilities can be estimated by maximum likelihood estimation. Yet the phrases most likely to contribute important translational and ordering information—the longest ones—are the ones most subject to sparse data issues.

Alternately, conditional phrasal models can be constructed from word translation probabilities; this approach is often called lexical weighting (Vogel et al. 2003). This avoids sparse data issues, but tends to prefer literal translations where the word-for-word probabilities are high Furthermore most approaches model phrases as bags of words, and fail to distinguish between local re-ordering possibilities.

Partitioning limitation
A phrasal approach partitions the sentence into strings of words, making several questionable assumptions along the way. First, the probability of the partitioning is never considered. Long phrases tend to be rare and therefore have sharp probability distributions. This adds an inherent bias toward long phrases with questionable MLE probabilities (e.g. 1/1 or 2/2). [1]

Second, the translation probability of each phrase pair is modeled independently. Such an approach fails to model any phenomena that reach across boundaries; only the target language model and perhaps whole-sentence bag of words models cross phrase boundaries. This is especially important when translating into languages with agreement phenomena. Often a single phrase does not cover all agreeing modifiers of a headword; the uncovered modifiers are biased toward the most common variant rather than the one agreeing with its head. Ideally a system would consider overlapping phrases rather than a single partitioning, but this poses a problem for generative models: when words are generated multiple times by different phrases, they are effectively penalized.

1.3. Practical problem with phrases: size

In addition to the theoretical problems with phrases, there are also practical issues. While phrasal systems achieve diminishing returns due

[1] The Alignment Template approach differs slightly here. Phrasal SMT estimates the probability of a phrase pair as:

$$\phi(\bar{t} \mid \bar{s}) = \frac{count(\bar{s}, \bar{t})}{\sum_{\bar{t}'} count(\bar{s}, \bar{t}')}$$

The Alignment Template method incorporates a loose partitioning probability by instead estimating the probability as (in the special case where each word has a unique class):

$$p(\bar{t} \mid \bar{s}) = \frac{count(\bar{s}, \bar{t})}{count(\bar{s})}$$

Note that these counts could differ significantly. Picture a source phrase that almost always translates into a discontiguous phrase (e.g. English *not* becoming French *ne … pas*), except for the rare occasion where, due to an alignment error or odd training data, it translates into a contiguous phrase (e.g. French *ne parle pas*). Then the first probability formulation of *ne parle pas* given *not* would be unreasonably high. However, this is a partial fix since it again suffers from data sparsity problems, especially on longer templates where systems hope to achieve the best benefits from phrases.

to sparse data, one does see a small incremental benefit with increasing phrase lengths. Given that storing all of these phrases leads to very large phrase tables, many research systems simply limit the phrases gathered to those that could possibly influence some test set. However, this is not feasible for true production MT systems, since the data to be translated is unknown.

2. Previous work

2.1. Delayed phrase construction

To avoid the major practical problem of phrasal SMT—namely large phrase tables, most of which are not useful to any one sentence—one can instead construct phrase tables on the fly using an indexed form of the training data (Zhang and Vogel 2005; Callison-Burch et al. 2005). However, this does not relieve any of the theoretical problems with phrase-based SMT.

2.2. Syntax-based SMT

Two recent systems have attempted to address the contiguity limitation and global re-ordering problem using syntax-based approaches.

Hierarchical phrases

Recent work in the use of hierarchical phrases (Chiang 2005) improves the ability to capture linguistic generalizations, and also removes the limitation to contiguous phrases. Hierarchical phrases differ from standard phrases in one important way: in addition to lexical items, a phrase pair may contain indexed placeholders, where each index must occur exactly once on each side. Such a formulation leads to a formally syntax-based translation approach, where translation is viewed as a parallel parsing problem over a grammar with one non-terminal symbol. This approach significantly outperforms a phrasal SMT baseline in controlled experimentation.

Hierarchical phrases do address the need for non-contiguous phrases and suggest a powerful ordering story in the absence of linguistic information, although this reordering information is bound in a deeply lexicalized form. Yet they do not address the phrase probability estimation problem; nor do they provide a means of modeling phenomena across phrase boundaries. The practical problems with phrase-based translation systems are further exacerbated, since the number of translation rules with up to two non-adjacent non-terminals in a 1-1 monotone sentence pair of n source and target words is $O(n^6)$, as compared to $O(n^2)$ phrases.

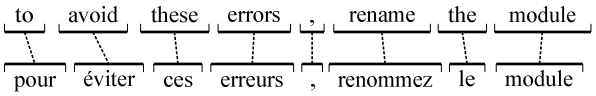
Treelet Translation

Another means of extending phrase-based translation is to incorporate source language syntactic information. In Quirk and Menezes (2005) we presented an approach to phrasal SMT based on a parsed dependency tree representation of the source language. We use a source dependency parser and project a target dependency tree using a word-based alignment, after which we extract tree-based phrases ('treelets') and train a tree-based ordering model. We showed that using treelets and a tree-based ordering model results in significantly better translations than a leading phrase-based system (Pharaoh, Koehn 2004), keeping all other models identical.

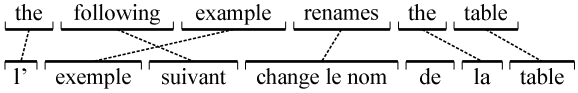
Like the hierarchical phrase approach, treelet translation succeeds in improving the global re-ordering search and allowing discontiguous phrases, but does not solve the partitioning or estimation problems. While we found our treelet system more resistant to degradation at smaller phrase sizes than the phrase-based system, it nevertheless suffered significantly at very small phrase sizes. Thus it is also subject to practical problems of size, and again these problems are exacerbated since there are potentially an exponential number of treelets.

2.3. Bilingual *n*-gram channel models

To address on the problems of estimation and partitioning, one recent approach transforms channel modeling into a standard sequence modeling problem (Banchs et al. 2005). Consider the following aligned sentence pair in Figure 1a. In such a well-behaved example, it is natural to consider the problem in terms of sequence models. Picture a generative process that produces a sentence pair in left to right, emitting a pair of words in lock step. Let $M = \langle m_1, \ldots, m_n \rangle$ be a sequence of word pairs $m_i = \langle s, t \rangle$. Then one can generatively model the probability of an aligned sentence pair using techniques from *n*-gram language modeling:



(a) Monotone aligned sentence pair



(b) More common non-monotone aligned sentence pair

Figure 1. Example aligned sentence pairs.

$$P(S,T,A) = P(M) = \sum_{i=1}^{k} P(m_i \mid m_1^{i-1}) \approx \sum_{i=1}^{k} P(m_i \mid m_{i-n}^{i-1})$$

When an alignment is one-to-one and monotone, this definition is sufficient. However alignments are seldom purely one-to-one and monotone in practice; Figure 1b displays common behavior such as one-to-many alignments, inserted words, and non-monotone translation. To address these problems, Banchs et al. (2005) suggest defining tuples such that:

(1) the tuple sequence is monotone,
(2) there are no word alignment links between two distinct tuples,
(3) each tuple has a non-NULL source side, which may require that target words aligned to NULL are joined with their following word, and
(4) no smaller tuples can be extracted without violating these constraints.

Note that *M* is now a sequence of phrase pairs instead of word pairs. With this adjusted definition, even Figure 1b can be generated using the same process using the following tuples:

m_1 = ‹ *the, l'* ›
m_2 = ‹ *following example, exemple suivant* ›
m_3 = ‹ *renames, change le nom* ›
m_4 = ‹ *the, de la* ›
m_5 = ‹ *table, table* ›

There are several advantages to such an approach. First, it largely avoids the partitioning problem; instead of segmenting into potentially large phrases, the sentence is segmented into much smaller tuples, most often pairs of single words. Furthermore the failure to model a partitioning probability is much more defensible when the partitions are much smaller. Secondly, n-gram language model probabilities provide a robust means of estimating phrasal translation probabilities in context that models interactions between *all* adjacent tuples, obviating the need for overlapping mappings.

These tuple channel models still must address practical issues such as model size, though much work has been done to shrink language models with minimal impact to perplexity (e.g. Stolcke 1998), which these models could immediately leverage. Furthermore, these models do not address the contiguity problem or the global reordering problem.

3. Translation by MTUs

In this paper, we address all four theoretical problems using a novel combination of our syntactically-informed treelet approach (Quirk and Menezes 2005) and a modified version of bilingual *n*-gram channel models (Banchs et al. 2005). As in our previous work, we first parse the sentence into a dependency tree. After this initial parse, we use a global search to find a candidate that maximizes a log-linear model, where these candidates consist of a target word sequence annotated with a dependency structure, a word alignment, and a treelet decomposition.

We begin by exploring minimal translation units and the models that concern them.

3.1. Minimal Translation Units

Minimal Translation Units (MTUs) are related to the tuples of Banchs et al. (2005), but differ in several important respects. First, we relieve the restriction that the MTU sequence be monotone. This prevents spurious expansion of MTUs to incorporate adjacent context only to satisfy monotonicity. In the example, note that the previous algorithm would extract the tuple ‹*following example, exemple suivant*› even though the translations are mostly independent. Their partitioning is also context dependent: if the sentence did not contain the words *following* or *suivant*, then ‹ *example, exemple* › would be a single MTU. Secondly we drop the requirement that no MTU have a NULL source side. While some insertions can be modeled in terms of adjacent words, we believe more robust models can be obtained if we consider insertions as

		English	French	English	Japanese
Training	Sentences	300,000		500,000	
	Words	4,441,465	5,198,932	7,909,198	9,379,240
	Vocabulary	63,343	59,290	79,029	95,813
	Singletons	35,328	29,448	44,111	52,911
Development test	Sentences	200		200	
	Words	3,045	3,456	3,436	4,095
Test	Sentences	2,000		2,000	
	Words	30,010	34,725	35,556	3,855
	OOV rate	5.5%	4.6%	6.9%	6.8%

Table 4.1 Data characteristics

independent units. In the end our MTUs are defined quite simply as pairs of source and target word sets that follow the given constraints:

(1) there are no word alignment links between distinct MTUs, and
(2) no smaller MTUs can be extracted without violating the previous constraint.

Since our word alignment algorithm is able to produce one-to-one, one-to-many, many-to-one, one-to-zero, and zero-to-one translations, these act as our basic units. As an example, let us consider example (1) once again. Using this new algorithm, the MTUs would be:

m_1 = ‹ *the, l'* ›
m_2 = ‹ *following, suivant* ›
m_3 = ‹ *example, exemple* ›
m_4 = ‹ *renames, change le nom* ›
m_5 = ‹ NULL, *de* ›
m_6 = ‹ *the, la* ›
m_7 = ‹ *table, table* ›

A finer grained partitioning into MTUs further reduces the data sparsity and partitioning issues associated with phrases. Yet it poses issues in modeling translation: given a sequence of MTUs that does not have a monotone segmentation, how do we model the probability of an aligned translation pair? We propose several solutions, and use each in a log-linear combination of models.

First, one may walk the MTUs in source order, ignoring insertion MTUs altogether. Such a model is completely agnostic of the target word order; instead of generating an aligned source target pair, it generates a source sentence along with a bag of target phrases. This approach expends a great deal of modeling effort in regenerating the source sentence, which may not be altogether desirable, though it does condition on surrounding translations. Also, it can be evaluated on candidates before orderings are considered. This latter property may be useful in two-stage decoding strategies where translations are considered before orderings.

Secondly, one may walk the MTUs in target order, ignoring deletion MTUs. Where the source-order MTU channel model expends probability mass generating the source sentence, this model expends a probability mass generating the target sentence and therefore may be somewhat redundant with the target language model.

Finally, one may walk the MTUs in dependency tree order. Let us assume that in addition to an aligned source-target candidate pair, we have a dependency parse of the source side. Where the past models conditioned on surface adjacent MTUs, this model conditions on tree adjacent MTUs. Currently we condition only on the ancestor chain, where $parent_1(m)$ is the parent MTU of m, $parent_2(m)$ is the grandparent of m, and so on:

$$P(S,T,A) = P(M) \approx \prod_{m \in M} P(m \mid parent_1^{n-1}(m))$$

This model hopes to capture information completely distinct from the other two models, such as translational preferences contingent on the head, even in the presence of long distance dependencies. Note that it generates unordered dependency tree pairs.

All of these models can be trained from a parallel corpus that has been word aligned and the source side dependency parsed. We walk through each sentence extracting MTUs in source, target, and tree order. Standard n-gram language modeling tools can be used to train MTU language models.

3.2. Decoding

We employ a dependency tree-based beam search decoder to search the space of translations. First the input is parsed into a dependency tree

structure. For each input node in the dependency tree, an n-best list of candidates is produced. Candidates consist of a target dependency tree along with a treelet and word alignment. The decoder generally assumes phrasal cohesion: candidates covering a substring (not subsequence) of the input sentence produce a potential substring (not subsequence) of the final translation. In addition to allowing a DP / beam decoder, this allows us to evaluate string-based models (such as the target language model and the source and target order MTU n-gram models) on partial candidates. This decoder is unchanged from our previous work: the MTU n-gram models are simply incorporated as feature functions in the log-linear combination. In the experiments section the MTU models are referred to as model set (1).

3.3. Other translation models

Phrasal channel models

We can estimate traditional channel models using maximum likelihood or lexical weighting:

$$f_{\text{DirectMLE}}(S,T,A) = \prod_{(\sigma,\tau)\in treelets(A)} \frac{c(\sigma,\tau)}{c(\sigma,*)}$$

$$f_{\text{InverseMLE}}(S,T,A) = \prod_{(\sigma,\tau)\in treelets(A)} \frac{c(\sigma,\tau)}{c(*,\tau)}$$

$$f_{\text{DirectM1}}(S,T,A) = \prod_{(\sigma,\tau)\in treelets(A)} \prod_{t\in\tau} \sum_{s\in\sigma} p(t \mid s)$$

$$f_{\text{InverseM1}}(S,T,A) = \prod_{(\sigma,\tau)\in treelets(A)} \prod_{s\in\sigma} \sum_{t\in\tau} p(s \mid t)$$

We use word probability tables $p(t \mid s)$ and $p(s \mid t)$ estimated by IBM Model 1 (Brown et al. 1993). Such models can be built over phrases if used in a phrasal decoder or over treelets if used in a treelet decoder. These models are referred to as set (2).

Word-based models

A target language model using modified Kneser-Ney smoothing captures fluency; a word count feature offsets the target LM preference for shorter selections; and a treelet/phrase count helps bias toward translations using fewer phrases. These models are referred to as set (3).

$$f_{\text{targetLM}}(S,T,A) = \prod_{i=1}^{|T|} P(t_i \mid t_{i-n}^{i-1})$$

$$f_{\text{wordcount}}(S,T,A) = |T|$$

$$f_{\text{phrasecount}}(S,T,A) = |treelets(A)|$$

Syntactic models

As in Quirk and Menezes (2005), we include a linguistically-informed order model that predicts the head-relative position of each node independently, and a tree-based bigram target language model; these models are referred to as set (4).

$$f_{\text{order}}(S,T,A) = \prod_{t\in T} P(position(t) \mid S,T,A)$$

$$f_{\text{treeLM}}(S,T,A) = \prod_{t\in T} P(t \mid parent(t))$$

4. Experimental setup

We evaluate the translation quality of the system using the BLEU metric (Papineni et al., 02) under a variety of configurations. As an additional baseline, we compare against a phrasal SMT decoder, Pharaoh (Koehn et al. 2003).

4.1. Data

Two language pairs were used for this comparison: English to French, and English to Japanese. The data was selected from technical software documentation including software manuals and product support articles; Table 4.1 presents the major characteristics of this data.

4.2. Training

We parsed the source (English) side of the corpora using NLPWIN, a broad-coverage rule-based parser able to produce syntactic analyses at varying levels of depth (Heidorn 2002). For the purposes of these experiments we used a dependency tree output with part-of-speech tags and unstemmed surface words. Word alignments were produced by GIZA++ (Och and Ney 2003) with a standard training regimen of five iterations of Model 1, five iterations of the HMM Model, and five iterations of Model 4, in both directions. These alignments were combined heuristically as described in our previous work.

We then projected the dependency trees and used the aligned dependency tree pairs to extract treelet translation pairs, train the order model, and train MTU models. The target language models were trained using only the target side of the corpus. Finally we trained model weights by maximizing BLEU (Och 2003) and set decoder optimization parameters (*n*-best list size, timeouts

	EF	EJ
Phrasal decoder (Pharaoh)		
Model sets (2),(3)	45.8±2.0	32.9±0.9
Treelet decoder, without discontiguous mappings		
Model sets (2),(3)	45.1±2.1	33.2±0.9
Model sets (2),(3),(4)	48.4±2.0	34.8±0.9
Treelet decoder, with discontiguous mappings		
Model sets (2),(3)	46.4±2.1	34.3±0.9
Model sets (2),(3),(4)	48.7±2.1	34.9±0.9
Model sets (1),(3),(4)	49.6±2.1	33.9±0.8
Model sets (1)-(4)	50.5±2.1	36.2±0.9

Table 5.1. Broad system comparison.

	EF	EJ
Treelet decoder, model sets (1),(3),(4)		
MTU unigram	47.8±2.1	33.2±0.9
MTU bigram	49.6±2.1	33.9±0.8
MTU trigram	49.9±2.0	34.0±0.9
MTU 4-gram	49.6±2.1	34.1±0.9
Treelet decoder, model sets (1)-(4)		
MTU unigram	48.6±2.1	34.3±1.0
MTU bigram	50.5±2.1	36.2±0.9
MTU trigram	48.9±2.0	36.1±0.9
MTU 4-gram	50.4±2.0	36.2±1.0

Table 5.2. Varying MTU n-gram model order.

etc) on a development test set of 200 held-out sentences each with a single reference translation. Parameters were individually estimated for each distinct configuration.

Pharaoh

The same GIZA++ alignments as above were used in the Pharaoh decoder (Koehn 2004). We used the heuristic combination described in (Och and Ney 2003) and extracted phrasal translation pairs from this combined alignment as described in (Koehn et al., 2003). Aside from MTU models and syntactic models (Pharaoh uses its own ordering approach), the same models were used: MLE and lexical weighting channel models, target LM, and phrase and word count. Model weights were also trained following Och (2003).

5. Results

We begin with a broad brush comparison of systems in Table 5.1. Throughout this section, treelet and phrase sizes are measured in terms of MTUs, not words. By default, all systems (including Pharaoh) use treelets or phrases of up to four MTUs, and MTU bigram models. The first results reiterate that the introduction of discontiguous mappings and especially a linguistically motivated order model (model set (4)) can improve translation quality. Replacing the standard channel models (model set (2)) with MTU bigram models (model set (1)) does not appear to degrade quality; it even seems to boost quality on EF. Furthermore, the information in the MTU models appears somewhat orthogonal to the phrasal models; a combination results in improvements for both language pairs.

The experiments in Table 5.2 compare quality using different orders of MTU n-gram models. (Treelets containing up to four MTUs were still used as the basis for decoding; only the order of the MTU n-gram models was adjusted.) A unigram model performs surprisingly well. This supports our intuition that atomic handling of non-compositional multi-word translations is a major contribution of phrasal SMT. Furthermore bigram models increase translation quality supporting the claim that local context is another contribution. Models beyond bigrams had little impact presumably due to sparsity and smoothing.

Table 5.3 explores the impact of using different phrase/treelet sizes in decoding. We see that adding MTU models makes translation more resilient given smaller phrases. The poor performance at size 1 is not particularly surprising: both systems require insertions to be lexically anchored: the only decoding operation allowed is translation of some visible source phrase, and insertions have no visible trace.

6. Conclusions

In this paper we have teased apart the role of

Table 5.3. Varying phrase / treelet size.

	Phrasal decoder model sets (2),(3)		*Treelet decoder: MTU bigram model sets (1),(3),(4)*		*Treelet decoder: MTU bigram model sets (1)-(4)*	
Size	**EF**	**EJ**	**EF**	**EJ**	**EF**	**EJ**
1	32.6±1.8	20.5±0.7	26.3±1.3	15.4±0.7	29.8±1.4	16.7±0.7
2	40.4±1.9	29.7±0.7	48.7±2.1	32.4±0.9	47.7±2.1	33.8±0.8
3	44.3±2.1	30.7±0.9	48.5±2.0	34.6±0.9	48.5±2.0	35.1±0.9
4	45.8±2.0	32.9±0.9	49.6±2.1	33.9±0.8	50.5±2.1	36.2±0.9

phrases and handled each contribution via a distinct model best suited to the task. Non-compositional translations stay as MTU phrases. Context and robust estimation is provided by MTU-based n-gram models. Local and global ordering is handled by a tree-based model.

The first interesting result is that at normal phrase sizes, augmenting an SMT system with MTU n-gram models improves quality; whereas replacing the standard phrasal channel models by the more theoretically sound MTU n-gram channel models leads to very similar performance.

Even more interesting are the results on smaller phrases. A system using very small phrases (size 2) and MTU bigram models matches (English-French) or at least approaches (English-Japanese) the performance of the baseline system using large phrases (size 4). While this work does not yet obviate the need for phrases, we consider it a promising step in that direction.

An immediate practical benefit is that it allows systems to use much smaller phrases (and hence smaller phrase tables) with little or no loss in quality. This result is particularly important for syntax-based systems, or any system that allows discontiguous phrases. Given a fixed length limit, the number of surface phrases extracted from any sentence pair of length n where all words are uniquely aligned is $O(n)$, but the number of treelets is potentially exponential in the number of children; and the number of rules with two gaps extracted by Chiang (2005) is potentially $O(n^3)$. Our results using MTUs suggest that such systems can avoid unwieldy, poorly estimated long phrases and instead anchor decoding on shorter, more tractable knowledge units such as MTUs, incorporating channel model information and contextual knowledge with an MTU n-gram model.

Much future work does remain. From inspecting the model weights of the best systems, we note that only the source order MTU n-gram model has a major contribution to the overall score of a given candidate. This suggests that the three distinct models, despite their different walk orders, are somewhat redundant. We plan to consider other approaches for conditioning on context. Furthermore phrasal channel models, in spite of the laundry list of problems presented here, have a significant impact on translation quality. We hope to replace them with effective models without the brittleness and sparsity issues of heavy lexicalization.

References

Banchs, Rafael, Josep Crego, Adrià de Gispert, Patrik Lambert, and Jose Mariño. 2005. Statistical machine translation of Euparl data by using bilingual n-grams. In *Proceedings of ACL Workshop on Building and Using Parallel Texts.*

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

Callison-Burch, Chris, Colin Bannard, and Josh Schroeder. 2005. Scaling phrase-based machine translation to larger corpora and longer phrases. In *Proceedings of ACL*.

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

Heidorn, George. 2000. "Intelligent writing assistance". In Dale et al. *Handbook of Natural Language Processing*, Marcel Dekker.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase based translation. In *Proceedings of NAACL*.

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

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

Och, Franz Josef and Hermann Ney. 2004. The Alignment Template approach to statistical machine translation, *Computational Linguistics*, 30(4):417-450.

Och, Franz Josef. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*.

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

Quirk, Chris and Arul Menezes. 2005. Dependency tree translation: syntactically-informed phrasal SMT. In *Proceedings of ACL*.

Stolcke, Andreas. 1998. Entropy-based pruning of backoff language models. In *Proceedings of DARPA Broadcast News Transcription and Understanding*.

Vogel, Stephan, Ying Zhang, Fei Huang, Alicia Tribble, Ashish Venugopal, Bing Zhao, Alex Waibel. 2003. The CMU statistical machine translation system. In *Proceedings of MT Summit.*

Zens, Richard, and Hermann Ney. 2003. A comparative study on reordering constraints in statistical machine translation. In *Proceedings of ACL*.

Zhang, Ying and Stephan Vogel. 2005. An efficient phrase-to-phrase alignment model for arbitrarily long phrase and large corpora. In *Proceedings of EAMT*.

Improved Statistical Machine Translation Using Paraphrases

Chris Callison-Burch Philipp Koehn Miles Osborne
School of Informatics
University of Edinburgh
2 Buccleuch Place
Edinburgh, EH8 9LW
callison-burch@ed.ac.uk

Abstract

Parallel corpora are crucial for training SMT systems. However, for many language pairs they are available only in very limited quantities. For these language pairs a huge portion of phrases encountered at run-time will be unknown. We show how techniques from paraphrasing can be used to deal with these otherwise unknown source language phrases. Our results show that augmenting a state-of-the-art SMT system with paraphrases leads to significantly improved coverage and translation quality. For a training corpus with 10,000 sentence pairs we increase the coverage of unique test set unigrams from 48% to 90%, with more than half of the newly covered items accurately translated, as opposed to none in current approaches.

1 Introduction

As with many other statistical natural language processing tasks, statistical machine translation (Brown et al., 1993) produces high quality results when ample training data is available. This is problematic for so called "low density" language pairs which do not have very large parallel corpora. For example, when words occur infrequently in a parallel corpus parameter estimates for word-level alignments can be inaccurate, which can in turn lead to inaccurate phrase translations. Limited amounts of training data can further lead to a problem of low coverage in that many phrases encountered at run-time are not observed in the training data and therefore their translations will not be learned.

Here we address the problem of unknown phrases. Specifically we show that upon encountering an unknown source phrase, we can substitute a paraphrase for it and then proceed using the translation of that paraphrase. We derive these paraphrases from resources that are external to the parallel corpus that the translation model is trained from, and we are able to exploit (potentially more abundant) parallel corpora from other language pairs to do so.

In this paper we:

- Define a method for incorporating paraphrases of unseen source phrases into the statistical machine translation process.
- Show that by translating paraphrases we achieve a marked improvement in coverage and translation quality, especially in the case of unknown words which to date have been left untranslated.
- Argue that while we observe an improvement in Bleu score, this metric is particularly poorly suited to measuring the sort of improvements that we achieve.
- Present an alternative methodology for targeted manual evaluation that may be useful in other research projects.

2 The Problem of Coverage in SMT

Statistical machine translation made considerable advances in translation quality with the introduction of phrase-based translation (Marcu and Wong, 2002; Koehn et al., 2003; Och and Ney, 2004). By

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 17–24,
New York, June 2006.

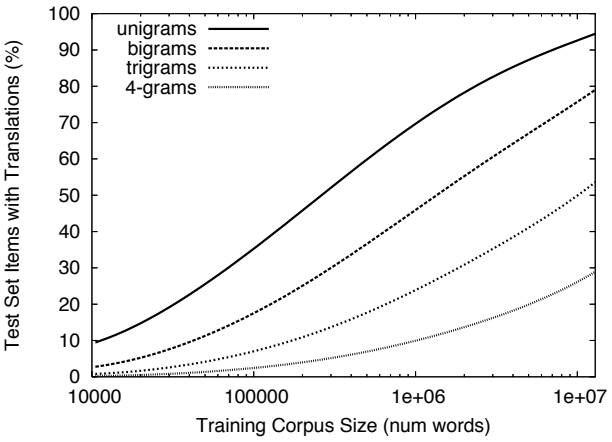


Figure 1: Percent of unique unigrams, bigrams, trigrams, and 4-grams from the Europarl Spanish test sentences for which translations were learned in increasingly large training corpora

increasing the size of the basic unit of translation, phrase-based machine translation does away with many of the problems associated with the original word-based formulation of statistical machine translation (Brown et al., 1993). For instance, with multi-word units less re-ordering needs to occur since local dependencies are frequently captured. For example, common adjective-noun alternations are memorized. However, since this linguistic information is not explicitly and generatively encoded in the model, unseen adjective noun pairs may still be handled incorrectly.

Thus, having observed phrases in the past dramatically increases the chances that they will be translated correctly in the future. However, for any given test set, a huge amount of training data has to be observed before translations are learned for a reasonable percentage of the test phrases. Figure 1 shows the extent of this problem. For a training corpus containing 10,000 words translations will have been learned for only 10% of the unigrams (*types*, not tokens). For a training corpus containing 100,000 words this increases to 30%. It is not until nearly 10,000,000 words worth of training data have been analyzed that translation for more than 90% of the vocabulary items have been learned. This problem is obviously compounded for higher-order n-grams (longer phrases), and for morphologically richer languages.

encargarnos	to ensure, take care, ensure that
garantizar	guarantee, ensure, guaranteed, assure, provided
velar	ensure, ensuring, safeguard, making sure
procurar	ensure that, try to, ensure, endeavour to
asegurarnos	ensure, secure, make certain
usado	used
utilizado	used, use, spent, utilized
empleado	used, spent, employee
uso	use, used, usage
utiliza	used, uses, used, being used
utilizar	to use, use, used

Table 1: Example of automatically generated paraphrases for the Spanish words *encargarnos* and *usado* along with their English translations which were automatically learned from the Europarl corpus

2.1 Handling unknown words

Currently most statistical machine translation systems are simply unable to handle unknown words. There are two strategies that are generally employed when an unknown source word is encountered. Either the source word is simply omitted when producing the translation, or alternatively it is passed through untranslated, which is a reasonable strategy if the unknown word happens to be a name (assuming that no transliteration need be done). Neither of these strategies is satisfying.

2.2 Using paraphrases in SMT

When a system is trained using 10,000 sentence pairs (roughly 200,000 words) there will be a number of words and phrases in a test sentence which it has not learned the translation of. For example, the Spanish sentence

> *Es positivo llegar a un acuerdo sobre los procedimientos, pero debemos encargarnos de que este sistema no sea susceptible de ser usado como arma política.*

may translate as

> It is good reach an agreement on procedures, but we must *encargarnos* that this system is not susceptible to be *usado* as political weapon.

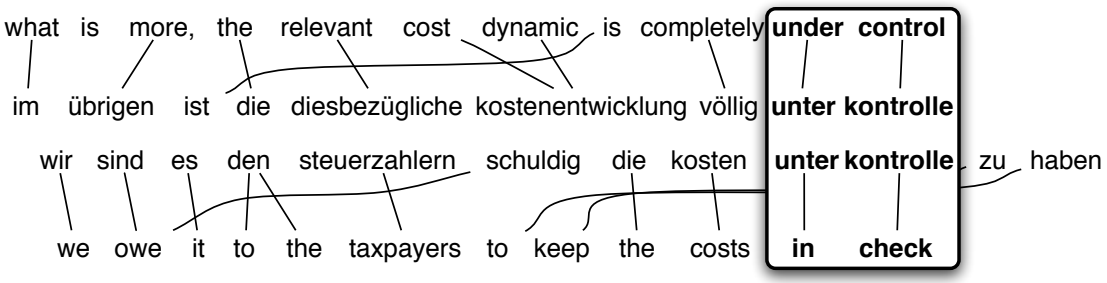


Figure 2: Using a bilingual parallel corpus to extract paraphrases

The strategy that we employ for dealing with unknown source language words is to substitute paraphrases of those words, and then translate the paraphrases. Table 1 gives examples of paraphrases and their translations. If we had learned a translation of *garantizar* we could translate it instead of *encargarnos*, and similarly for *utilizado* instead of *usado*.

3 Acquiring Paraphrases

Paraphrases are alternative ways of expressing the same information within one language. The automatic generation of paraphrases has been the focus of a significant amount of research lately. Many methods for extracting paraphrases (Barzilay and McKeown, 2001; Pang et al., 2003) make use of monolingual parallel corpora, such as multiple translations of classic French novels into English, or the multiple reference translations used by many automatic evaluation metrics for machine translation.

Bannard and Callison-Burch (2005) use bilingual parallel corpora to generate paraphrases. Paraphrases are identified by pivoting through phrases in another language. The foreign language translations of an English phrase are identified, all occurrences of those foreign phrases are found, and all English phrases that they translate back to are treated as potential paraphrases of the original English phrase. Figure 2 illustrates how a German phrase can be used as a point of identification for English paraphrases in this way.

The method defined in Bannard and Callison-Burch (2005) has several features that make it an ideal candidate for incorporation into statistical machine translation system. Firstly, it can easily be applied to any language for which we have one or more parallel corpora. Secondly, it defines a paraphrase probability, $p(e_2|e_1)$, which can be incorporated into the probabilistic framework of SMT.

3.1 Paraphrase probabilities

The paraphrase probability $p(e_2|e_1)$ is defined in terms of two translation model probabilities: $p(f|e_1)$, the probability that the original English phrase e_1 translates as a particular phrase f in the other language, and $p(e_2|f)$, the probability that the candidate paraphrase e_2 translates as the foreign language phrase. Since e_1 can translate as multiple foreign language phrases, we marginalize f out:

$$p(e_2|e_1) = \sum_f p(f|e_1)p(e_2|f) \quad (1)$$

The translation model probabilities can be computed using any standard formulation from phrase-based machine translation. For example, $p(e_2|f)$ can be calculated straightforwardly using maximum likelihood estimation by counting how often the phrases e and f were aligned in the parallel corpus:

$$p(e_2|f) \approx \frac{count(e_2, f)}{\sum_{e_2} count(e_2, f)} \quad (2)$$

There is nothing that limits us to estimating paraphrases probabilities from a single parallel corpus. We can extend the definition of the paraphrase probability to include multiple corpora, as follows:

$$p(e_2|e_1) \approx \frac{\sum_{c \in C} \sum_{f \text{ in } c} p(f|e_1)p(e_2|f)}{|C|} \quad (3)$$

where c is a parallel corpus from a set of parallel corpora C. Thus multiple corpora may be used

by summing over all paraphrase probabilities calculated from a single corpus (as in Equation 1) and normalized by the number of parallel corpora.

4 Experimental Design

We examined the application of paraphrases to deal with unknown phrases when translating from Spanish and French into English. We used the publicly available Europarl multilingual parallel corpus (Koehn, 2005) to create six training corpora for the two language pairs, and used the standard Europarl development and test sets.

4.1 Baseline

For a baseline system we produced a phrase-based statistical machine translation system based on the log-linear formulation described in (Och and Ney, 2002)

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}) \quad (4)$$

$$= \arg\max_{\mathbf{e}} \sum_{m=1}^{M} \lambda_m h_m(\mathbf{e}, \mathbf{f}) \quad (5)$$

The baseline model had a total of eight feature functions, $h_m(\mathbf{e}, \mathbf{f})$: a language model probability, a phrase translation probability, a reverse phrase translation probability, lexical translation probability, a reverse lexical translation probability, a word penalty, a phrase penalty, and a distortion cost. To set the weights, λ_m, we performed minimum error rate training (Och, 2003) on the development set using Bleu (Papineni et al., 2002) as the objective function.

The phrase translation probabilities were determined using maximum likelihood estimation over phrases induced from word-level alignments produced by performing Giza++ training on each of the three training corpora. We used the Pharaoh beam-search decoder (Koehn, 2004) to produce the translations after all of the model parameters had been set.

When the baseline system encountered unknown words in the test set, its behavior was simply to reproduce the foreign word in the translated output. This is the default behavior for many systems, as noted in Section 2.1.

4.2 Translation with paraphrases

We extracted all source language (Spanish and French) phrases up to length 10 from the test and development sets which did not have translations in phrase tables that were generated for the three training corpora. For each of these phrases we generated a list of paraphrases using all of the parallel corpora from Europarl aside from the Spanish-English and French-English corpora. We used bitexts between Spanish and Danish, Dutch, Finnish, French, German, Italian, Portuguese, and Swedish to generate our Spanish paraphrases, and did similarly for the French paraphrases. We manage the parallel corpora with a suffix array -based data structure (Callison-Burch et al., 2005). We calculated paraphrase probabilities using the Bannard and Callison-Burch (2005) method, summarized in Equation 3. Source language phrases that included names and numbers were not paraphrased.

For each paraphrase that had translations in the phrase table, we added additional entries in the phrase table containing the original phrase and the paraphrase's translations. We augmented the baseline model by incorporating the paraphrase probability into an additional feature function which assigns values as follows:

$$h(\mathbf{e}, \mathbf{f_1}) = \begin{cases} p(\mathbf{f_2}|\mathbf{f_1}) & \text{If phrase table entry } (\mathbf{e}, \mathbf{f_1}) \\ & \text{is generated from } (\mathbf{e}, \mathbf{f_2}) \\ 1 & \text{Otherwise} \end{cases}$$

Just as we did in the baseline system, we performed minimum error rate training to set the weights of the nine feature functions in our translation model that exploits paraphrases.

We tested the usefulness of the paraphrase feature function by performing an additional experiment where the phrase table was expanded but the paraphrase probability was omitted.

4.3 Evaluation

We evaluated the efficacy of using paraphrases in three ways: by calculating the Bleu score for the translated output, by measuring the increase in coverage when including paraphrases, and through a targeted manual evaluation of the phrasal translations of unseen phrases to determine how many of the newly covered phrases were accurately translated.

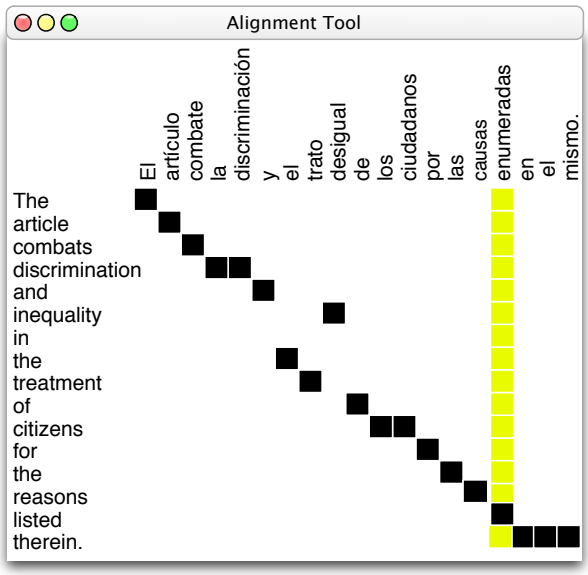


Figure 3: Test sentences and reference translations were manually word-aligned. This allowed us to equate unseen phrases with their corresponding English phrase. In this case *enumeradas* with *listed*.

The article combats discrimination and inequality in the treatment of citizens for the reasons **listed** therein.
The article combats discrimination and the different treatment of citizens for the reasons **mentioned** in the same.
The article fights against uneven and the treatment of citizens for the reasons **enshrined** in the same.
The article is countering discrimination and the unequal treatment of citizens for the reasons **that** in the same.

Figure 4: Judges were asked whether the highlighted phrase retained the same meaning as the highlighted phrase in the reference translation (top)

Although Bleu is currently the standard metric for MT evaluation, we believe that it may not meaningfully measure translation improvements in our setup. By substituting a paraphrase for an unknown source phrase there is a strong chance that its translation may also be a paraphrase of the equivalent target language phrase. Bleu relies on exact matches of n-grams in a reference translation. Thus if our translation is a paraphrase of the reference, Bleu will fail to score it correctly.

Because Bleu is potentially insensitive to the type of changes that we were making to the translations, we additionally performed a focused manual evaluation (Callison-Burch et al., 2006). To do this, had bilingual speakers create word-level alignments for the first 150 and 250 sentence in the Spanish-English and French-English test corpora, as shown in Figure 3. We were able to use these alignments to extract the translations of the Spanish and French words that we were applying our paraphrase method to.

Knowing this correspondence between foreign phrases and their English counterparts allowed us to directly analyze whether translations that were being produced from paraphrases remained faithful to the meaning of the reference translation. When producing our translations using the Pharaoh decoder we employed its "trace" facility, which tells which source sentence span each target phrase was derived from. This allowed us to identify which elements in the machine translated output corresponded to the paraphrased foreign phrase. We asked a monolingual judge whether the phrases in the machine translated output had the same meaning as of the reference phrase. This is illustrated in Figure 4.

In addition to judging the accuracy of 100 phrases for each of the translated sets, we measured how much our paraphrase method increased the coverage of the translation system. Because we focus on words that the system was previously unable to translate, the increase in coverage and the translation quality of the newly covered phrases are the two most relevant indicators as to the efficacy of the method.

5 Results

We produced translations under five conditions for each of our training corpora: a set of baseline translations without any additional entries in the phrase table, a condition where we added the translations of paraphrases for unseen source words along with paraphrase probabilities, a condition where we added the translations of paraphrases of multi-word phrases along with paraphrase probabilities, and two additional conditions where we added the translations of paraphrases of single and multi-word paraphrase without paraphrase probabilities.

	Spanish-English						French-English					
Corpus size	10k	20k	40k	80k	160k	320k	10k	20k	40k	80k	160k	320k
Baseline	22.6	25.0	26.5	26.5	28.7	**30.0**	21.9	24.3	26.3	27.8	28.8	29.5
Single word	23.1	25.2	26.6	**28.0**	**29.0**	**30.0**	22.7	24.2	26.9	27.7	28.9	**29.8**
Multi-word	**23.3**	**26.0**	**27.2**	**28.0**	28.8	29.7	**23.7**	**25.1**	**27.1**	**28.5**	**29.1**	**29.8**

Table 2: Bleu scores for the various training corpora, including baseline results without paraphrasing, results for only paraphrasing unknown words, and results for paraphrasing any unseen phrase. Corpus size is measured in sentences.

Corpus size	10k	20k	40k	80k	160k	320k	10k	20k	40k	80k	160k	320k
Single w/o-ff	23.0	25.1	26.7	28.0	**29.0**	29.9	22.5	24.1	26.0	27.6	28.8	29.6
Multi w/o-ff	20.6	22.6	21.9	24.0	25.4	27.5	19.7	22.1	24.3	25.6	26.0	28.1

Table 3: Bleu scores for the various training corpora, when the paraphrase feature function *is not* included

5.1 Bleu scores

Table 2 gives the Bleu scores for each of these conditions. We were able to measure a translation improvement for all sizes of training corpora, under both the single word and multi-word conditions, except for the largest Spanish-English corpus. For the single word condition, it would have been surprising if we had seen a decrease in Bleu score. Because we are translating words that were previously untranslatable it would be unlikely that we could do any worse. In the worst case we would be replacing one word that did not occur in the reference translation with another, and thus have no effect on Bleu.

More interesting is the fact that by paraphrasing unseen multi-word units we get an increase in quality above and beyond the single word paraphrases. These multi-word units may not have been observed in the training data as a unit, but each of the component words may have been. In this case translating a paraphrase would not be guaranteed to received an improved or identical Bleu score, as in the single word case. Thus the improved Bleu score is notable.

Table 3 shows that incorporating the paraphrase probability into the model's feature functions plays a critical role. Without it, the multi-word paraphrases harm translation performance when compared to the baseline.

5.2 Manual evaluation

We performed a manual evaluation by judging the accuracy of phrases for 100 paraphrased translations from each of the sets using the manual word alignments.[1] Table 4 gives the percentage of time that each of the translations of paraphrases were judged to have the same meaning as the equivalent target phrase. In the case of the translations of single word paraphrases for the Spanish accuracy ranged from just below 50% to just below 70%. This number is impressive in light of the fact that none of those items are correctly translated in the baseline model, which simply inserts the foreign language word. As with the Bleu scores, the translations of multi-word paraphrases were judged to be more accurate than the translations of single word paraphrases.

In performing the manual evaluation we were additionally able to determine how often Bleu was capable of measuring an actual improvement in translation. For those items judged to have the same meaning as the gold standard phrases we could track how many would have contributed to a higher Bleu score (that is, which of them were exactly the same as the reference translation phrase, or had some words in common with the reference translation phrase). By counting how often a correct phrase would have contributed to an increased Bleu score, and how often it would fail to increase the Bleu score we were able to determine with what frequency Bleu was sensitive to our improvements. We found that Bleu was insensitive to our translation improvements between 60-75% of the time, thus re-

[1] Note that for the larger training corpora fewer than 100 paraphrases occurred in the first 150 and 250 sentence pairs.

	Spanish-English						French-English					
Corpus size	10k	20k	40k	80k	160k	320k	10k	20k	40k	80k	160k	320k
Single word	48%	53%	57%	67%*	33%*	50%*	54%	49%	45%	50%	39%*	21%*
Multi-word	64%	65%	66%	71%	76%*	71%*	60%	67%	63%	58%	65%	42%*

Table 4: Percent of time that the translation of a paraphrase was judged to retain the same meaning as the corresponding phrase in the gold standard. Starred items had fewer than 100 judgments and should not be taken as reliable estimates.

Size	1-gram	2-gram	3-gram	4-gram
10k	48%	25%	10%	3%
20k	60%	35%	15%	6%
40k	71%	45%	22%	9%
80k	80%	55%	29%	12%
160k	86%	64%	37%	17%
320k	91%	71%	45%	22%

Table 5: The percent of the unique test set phrases which have translations in each of the Spanish-English training corpora prior to paraphrasing

Size	1-gram	2-gram	3-gram	4-gram
10k	90%	67%	37%	16%
20k	90%	69%	39%	17%
40k	91%	71%	41%	18%
80k	92%	73%	44%	20%
160k	92%	75%	46%	22%
320k	93%	77%	50%	25%

Table 6: The percent of the unique test set phrases which have translations in each of the Spanish-English training corpora after paraphrasing

inforcing our belief that it is not an appropriate measure for translation improvements of this sort.

5.3 Increase in coverage

As illustrated in Figure 1, translation models suffer from sparse data. When only a very small parallel corpus is available for training, translations are learned for very few of the unique phrases in a test set. If we exclude 451 words worth of names, numbers, and foreign language text in 2,000 sentences that comprise the Spanish portion of the Europarl test set, then the number of unique n-grams in text are: 7,331 unigrams, 28,890 bigrams, 44,194 trigrams, and 48,259 4-grams. Table 5 gives the percentage of these which have translations in each of the three training corpora, if we do not use paraphrasing.

In contrast after expanding the phrase table using the translations of paraphrases, the coverage of the unique test set phrases goes up dramatically (shown in Table 6). For the first training corpus with 10,000 sentence pairs and roughly 200,000 words of text in each language, the coverage goes up from less than 50% of the vocabulary items being covered to 90%. The coverage of unique 4-grams jumps from 3% to 16% – a level reached only after observing more than 100,000 sentence pairs, or roughly three million words of text, without using paraphrases.

6 Related Work

Previous research on trying to overcome data sparsity issues in statistical machine translation has largely focused on introducing morphological analysis as a way of reducing the number of types observed in a training text. For example, Nissen and Ney (2004) apply morphological analyzers to English and German and are able to reduce the amount of training data needed to reach a certain level of translation quality. Goldwater and McClosky (2005) find that stemming Czech and using lemmas improves the word-to-word correspondences when training Czech-English alignment models. Koehn and Knight (2003) show how monolingual texts and parallel corpora can be used to figure out appropriate places to split German compounds.

Still other approaches focus on ways of acquiring data. Resnik and Smith (2003) develop a method for gathering parallel corpora from the web. Oard et al. (2003) describe various methods employed for quickly gathering resources to create a machine translation system for a language with no initial resources.

7 Discussion

In this paper we have shown that significant gains in coverage and translation quality can be had by integrating paraphrases into statistical machine translation. In effect, paraphrases introduce some amount of *generalization* into statistical machine translation. Whereas before we relied on having observed a particular word or phrase in the training set in order to produce a translation of it, we are no longer tied to having seen every word in advance. We can exploit knowledge that is external to the translation model about what words have similar meanings and use that in the process of translation. This method is particularly pertinent to small data conditions, which are plagued by sparse data problems.

In future work, we plan to determine how much data is required to learn useful paraphrases. The scenario described in this paper was very favorable to creating high quality paraphrases. The large number of parallel corpora between Spanish and the other languages present in the Europarl corpus allowed us to generate high quality, in domain data. While this is a realistic scenario, in that many new official languages have been added to the European Union, some of which do not yet have extensive parallel corpora, we realize that this may be a slightly idealized scenario.

Finally, we plan to formalize our targeted manual evaluation method, in the hopes of creating a evaluation methodology for machine translation that is more thorough and elucidating than Bleu.

Acknowledgments

Thank you to Alexandra Birch and Stephanie Vandamme for creating the word alignments.

References

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *ACL-2005*.

Regina Barzilay and Kathleen McKeown. 2001. Extracting paraphrases from a parallel corpus. In *ACL-2001*.

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

Chris Callison-Burch, Colin Bannard, and Josh Schroeder. 2005. Scaling phrase-based statistical machine translation to larger corpora and longer phrases. In *Proceedings of ACL*.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of bleu in machine translation. In *Proceedings of EACL*.

Sharon Goldwater and David McClosky. 2005. Improving statistical MT through morphological analysis. In *Proceedings of EMNLP*.

Philipp Koehn and Kevin Knight. 2003. Empirical methods for compound splitting. In *Proceedings of EACL*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT/NAACL*.

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

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

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP*.

Sonja Nissen and Hermann Ney. 2004. Statistical machine translation with scarce resources using morpho-syntatic analysis. *Computational Linguistics*, 30(2):181–204.

Doug Oard, David Doermann, Bonnie Dorr, Daqing He, Phillip Resnik, William Byrne, Sanjeeve Khudanpur, David Yarowsky, Anton Leuski, Philipp Koehn, and Kevin Knight. 2003. Desperately seeking Cebuano. In *Proceedings of HLT-NAACL*.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of ACL*.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*.

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

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*.

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

Philip Resnik and Noah Smith. 2003. The web as a parallel corpus. *Computational Linguistics*, 29(3):349–380, September.

Segment Choice Models: Feature-Rich Models for Global Distortion in Statistical Machine Translation

Roland Kuhn, Denis Yuen, Michel Simard, Patrick Paul, George Foster, Eric Joanis, and Howard Johnson

Institute for Information Technology, National Research Council of Canada
Gatineau, Québec, CANADA
Email: {Roland.Kuhn, Michel.Simard, Patrick.Paul, George.Foster, Eric.Joanis, Howard.Johnson}@cnrc-nrc.gc.ca; Denis Yuen: mucous@gmail.com

Abstract

This paper presents a new approach to distortion (phrase reordering) in phrase-based machine translation (MT). Distortion is modeled as a sequence of choices during translation. The approach yields trainable, probabilistic distortion models that are global: they assign a probability to each possible phrase reordering. These "segment choice" models (SCMs) can be trained on "segment-aligned" sentence pairs; they can be applied during decoding or rescoring. The approach yields a metric called "distortion perplexity" ("disperp") for comparing SCMs offline on test data, analogous to perplexity for language models. A decision-tree-based SCM is tested on Chinese-to-English translation, and outperforms a baseline distortion penalty approach at the 99% confidence level.

1 Introduction: Defining SCMs

The work presented here was done in the context of phrase-based MT (Koehn *et al*., 2003; Och and Ney, 2004). Distortion in phrase-based MT occurs when the order of phrases in the source-language sentence changes during translation, so the order of corresponding phrases in the target-language translation is different. Some MT systems allow arbitrary reordering of phrases, but impose a distortion penalty proportional to the difference between the new and the original phrase order (Koehn, 2004). Some interesting recent research focuses on reordering within a narrow window of phrases (Kumar and Byrne, 2005; Tillmann and Zhang, 2005; Tillmann, 2004). The (Tillmann, 2004) paper introduced lexical features for distortion modeling. A recent paper (Collins *et al*., 2005) shows that major gains can be obtained by constructing a parse tree for the source sentence and then applying hand-crafted reordering rules to rewrite the source in target-language-like word order prior to MT.

Our model assumes that the source sentence is completely segmented prior to distortion. This simplifying assumption requires generation of hypotheses about the segmentation of the complete source sentence during decoding. The model also assumes that each translation hypothesis grows in a predetermined order. *E.g.*, Koehn's decoder (Koehn 2004) builds each new hypothesis by adding phrases to it left-to-right (order is deterministic for the target hypothesis). Our model doesn't require this order of operation – it would support right-to-left or inwards-outwards hypothesis construction – but it does require a predictable order.

One can keep track of how segments in the source sentence have been rearranged during decoding for a given hypothesis, using what we call a "distorted source-language hypothesis" (DSH). A similar concept appears in (Collins *et al*., 2005) (this paper's preoccupations strongly resemble

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 25–32,
New York, June 2006.

ours, though our method is completely different: we don't parse the source, and use only automatically generated rules). **Figure 1** shows an example of a DSH for German-to-English translation (case information is removed). Here, German "ich habe das buch gelesen ." is translated into English "i have read the book ." The DSH shows the distortion of the German segments into an English-like word order that occurred during translation (we tend to use the word "segment" rather than the more linguistically-charged "phrase").

Original German: **[ich]** [**habe]** [**das buch]** [**gelesen]** [**.]**
DSH for German: **[ich]** [**habe]** [**gelesen]** [**das buch]** [**.]**
(English: [*i*] [*have*] [*read*] [*the book*] [.]*)*

Figure 1. Example of German-to-English DSH

From the DSH, one can reconstruct the series of segment choices. In **Figure 1** - given a left-to-right decoder - "**[ich]**" was chosen from five candidates to be the leftmost segment in the DSH. Next, "**[habe]**" was chosen from four remaining candidates, "**[gelesen]**" from three candidates, and "**[das buch]**" from two candidates. Finally, the decoder was forced to choose "**[.]**".

Segment Choice Models (SCMs) assign probabilities to segment choices made as the DSH is constructed. The available choices at a given time are called the "Remaining Segments" (RS). Consider a valid (though stupid) SCM that assigns equal probabilities to all segments in the RS. This uniform SCM assigns a probability of $1/5!$ to the DSH in **Figure 1**: the probability of choosing "**[ich]**" from among 5 RS was $1/5$, then the probability of "**[habe]**" among 4 RS was $1/4$, *etc*. The uniform SCM would be of little use to an MT system. In the next two sections we describe some more informative SCMs, define the "distortion perplexity" ("disperp") metric for comparing SCMs offline on a test corpus, and show how to construct this corpus.

2 Disperp and Distortion Corpora

2.1 Defining Disperp

The ultimate reason for choosing one SCM over another will be the performance of an MT system containing it, as measured by a metric like BLEU (Papineni *et al.,* 2002). However, training and testing a large-scale MT system for each new SCM would be costly. Also, the distortion component's effect on the total score is muffled by other components (*e.g.*, the phrase translation and target language models). Can we devise a quick standalone metric for comparing SCMs?

There **is** an offline metric for statistical language models: perplexity (Jelinek, 1990). By analogy, the higher the overall probability a given SCM assigns to a test corpus of representative distorted sentence hypotheses (DSHs), the better the quality of the SCM. To define distortion perplexity ("disperp"), let $\mathbf{Pr_M(d_k)}$ = the probability an SCM **M** assigns to a DSH for sentence **k**, $\mathbf{d_k}$. If **T** is a test corpus comprising numerous DSHs, the probability of the corpus according to **M** is $\mathbf{Pr_M(T)} = \prod_k \mathbf{Pr_M(d_k)}$. Let **S(T)** = total number of segments in **T**. Then $\boldsymbol{disperp}\mathbf{(M,T)} = \mathbf{Pr_M(T)}^{-1/S(T)}$. This gives the mean number of choices model **M** allows; the lower the disperp for corpus **T**, the better **M** is as a model for **T** (a model **X** that predicts segment choice in **T** perfectly would have $\boldsymbol{disperp}\mathbf{(X,T)} = 1.0$).

2.2 Some Simple *A Priori* SCMs

The uniform SCM assigns to the DSH $\mathbf{d_k}$ that has $\mathbf{S(d_k)}$ segments the probability $\mathbf{1/[S(d_k)!]}$. We call this **Model A.** Let's define some other illustrative SCMs. **Fig. 2** shows a sentence that has 7 segments with 10 words (numbered 0-9 by original order). Three segments in the source have been used; the decoder has a choice of four RS. Which of the RS has the highest probability of being chosen? Perhaps **[2 3]**, because it is the leftmost RS: the "leftmost" predictor. Or, the last phrase in the DSH will be followed by the phrase that originally followed it, **[8 9]**: the "following" predictor. Or, perhaps positions in the source and target should be close, so since the next DSH position to be filled is 4, phrase **[4]** should be favoured: the "parallel" predictor.

original: [**0 1]** [**2 3]** [**4]** [**5]** [**6] [7] [8 9]**
DSH: **[0 1] [5] [7],** RS**: [2 3], [4], [6], [8 9]**

Figure 2. Segment choice prediction example

Model B will be based on the "leftmost" predictor, giving the leftmost segment in the RS twice the probability of the other segments, and giving the

others uniform probabilities. **Model C** will be based on the "following" predictor, doubling the probability for the segment in the RS whose first word was the closest to the last word in the DSH, and otherwise assigning uniform probabilities. Finally, **Model D** combines "leftmost" and "following": where the leftmost and following segments are different, both are assigned double the uniform probability; if they are the same segment, that segment has four times the uniform probability. Of course, the factor of 2.0 in these models is arbitrary. For **Figure 2**, probabilities would be:

- **Model A**: $Pr_A([2\ 3]) = Pr_A([4]) = Pr_A([6]) = Pr_A([8\ 9]) = 1/4$;
- **Model B**: $Pr_B([2\ 3]) = 2/5$, $Pr_B([4]) = Pr_B([6]) = Pr_B([8\ 9]) = 1/5$;
- **Model C**: $Pr_C([2\ 3]) = Pr_C([4]) = Pr_C([6]) = 1/5$, $Pr_C([8\ 9]) = 2/5$;
- **Model D**: $Pr_D([2\ 3]) = Pr_D([8\ 9]) = 1/3$, $Pr_D([4]) = Pr_D([6]) = 1/6$.

Finally, let's define an SCM derived from the distortion penalty used by systems based on the "following" predictor, as in (Koehn, 2004). Let a_i = start position of source phrase translated into *i*th target phrase, b_{i-1} = end position of source phrase that's translated into (*i-1*)th target phrase. Then distortion penalty $d(a_i, b_{i-1}) = \alpha^{|a_i - b_{i-1} - 1|}$; the total distortion is the product of the phrase distortion penalties. This penalty is applied as a kind of non-normalized probability in the decoder. The value of α for given (source, target) languages is optimized on development data.

To turn this penalty into an SCM, penalties are normalized into probabilities, at each decoding stage; we call the result **Model P** (for "penalty"). **Model P** with $\alpha = 1.0$ is the same as uniform **Model A**. In disperp experiments, **Model P** with α optimized on held-out data performs better than **Models A-D** (see **Figure 5**), suggesting that disperp is a realistic measure.

Models A-D are models whose parameters were all defined *a priori*; **Model P** has one trainable parameter, α. Next, let's explore distortion models with several trainable parameters.

2.3 Constructing a Distortion Corpus

To compare SCMs using disperp and to train complex SCMs, we need a corpus of representative examples of DSHs. There are several ways of obtaining such a corpus. For the experiments described here, the MT system was first trained on a bilingual sentence-aligned corpus. Then, the system was run in a second pass over its own training corpus, using its phrase table with the standard distortion penalty to obtain a best-fit phrase alignment between each (source, target) sentence pair. Each such alignment yields a DSH whose segments are aligned with their original positions in the source; we call such a source-DSH alignment a "segment alignment". We now use a leave-one-out procedure to ensure that information derived from a given sentence pair is not used to segment-align that sentence pair. In our initial experiments we didn't do this, with the result that the segment-aligned corpus underrepresented the case where words or N-grams not in the phrase table are seen in the source sentence during decoding.

3 A Trainable Decision Tree SCM

Almost any machine learning technique could be used to create a trainable SCM. We implemented one based on decision trees (DTs), not because DTs necessarily yield the best results but for software engineering reasons: DTs are a quick way to explore a variety of features, and are easily interpreted when grown (so that examining them can suggest further features). We grew N DTs, each defined by the number of choices available at a given moment. The highest-numbered DT has a "+" to show it handles N+1 or more choices. *E.g.*, if we set N=4, we grow a "2-choice", a "3-choice", a "4-choice", and a "5+-choice tree". The 2-choice tree handles cases where there are 2 segments in the RS, assigning a probability to each; the 3-choice tree handles cases where there are 3 segments in the RS, *etc*. The 5+-choice tree is different from the others: it handles cases where there are 5 segments in the RS to choose from, **and** cases where there are more than 5. The value of N is arbitrary; *e.g.*, for N=8, the trees go from "2-choice" up to "9+-choice".

Suppose a left-to-right decoder with an N=4 SCM is translating a sentence with seven phrases. Initially, when the DSH is empty, the 5+-choice tree assigns probabilities to each of these seven. It

will use the 5+-choice tree twice more, to assign probabilities to six RS, then to five. To extend the hypothesis, it will then use the 4-choice tree, the 3-choice tree, and finally the 2-choice tree. Disperps for this SCM are calculated on test corpus DSHs in the same left-to-right way, using the tree for the number of choices in the RS to find the probability of each segment choice.

Segments need labels, so the N-choice DT can assign probabilities to the N segments in the RS. We currently use a "following" labeling scheme. Let X be the original source position of the last word put into the DSH, plus 1. In **Figure 2**, this was word 7, so X=8. In our scheme, the RS segment whose first word is closest to X is labeled "A"; the second-closest segment is labeled "B", *etc*. Thus, segments are labeled in order of the (Koehn, 2004) penalty; the "A" segment gets the lowest penalty. Ties between segments on the right and the left of X are broken by first labeling the right segment. In **Figure 2**, the labels for the RS are "A" = [8 9], "B" = [6], "C" = [4], "D" = [2 3].

1. Position Questions
Segment Length Questions
E.g., "lgth(DSH)<5?", "lgth(B)=2?", "lgth(RS)<6?", *etc.*
Questions about Original Position
Let pos(seg) = index of seg's first word in source sentence
E.g., "pos(A)=9?", "pos(C) <17?", *etc.*
Questions With X ("following" word position)
E.g., "pos(X)=9?", "pos(C) – pos(X) <0?", *etc.*
Segment Order Questions
Let fseg = segment # (forward), bseg = segment # (backward)
E.g., "fseg(D) = 1?", "bseg(A) <5?", *etc.*
2. Word-Based Questions
E.g., "*and* ε DSH?", "*November* ε B?", *etc.*

Figure 3. Some question types for choice DTs

Figure 3 shows the main types of questions used for tree-growing, comprising **position questions** and **word-based questions**. Position questions pertain to location, length, and ordering of segments. Some position questions ask about the distance between the first word of a segment and the "following" position X: *e.g.*, if the answer to "pos(A)-pos(X)=0?" is yes, then segment A comes immediately after the last DSH segment in the source, and is thus highly likely to be chosen. There are also questions relating to the "leftmost" and "parallel" predictors (above, sec. 2.2). The fseg() and bseg() functions count segments in the RS from left to right and right to left respectively, allowing, *e.g.*, the question whether a given segment is the second last segment in the RS. The only word-based questions currently implemented ask whether a given word is contained in a given segment (or anywhere in the DSH, or anywhere in the RS). This type could be made richer by allowing questions about the position of a given word in a given segment, questions about syntax, *etc*.

Figure 4 shows an example of a 5+-choice DT. The "+" in its name indicates that it will handle cases where there are 5 or more segments in the RS. The counts stored in the leaves of this DT represent the number of training data items that ended up there; the counts are used to estimate probabilities. Some smoothing will be done to avoid zero probabilities, *e.g.*, for class C in node 3.

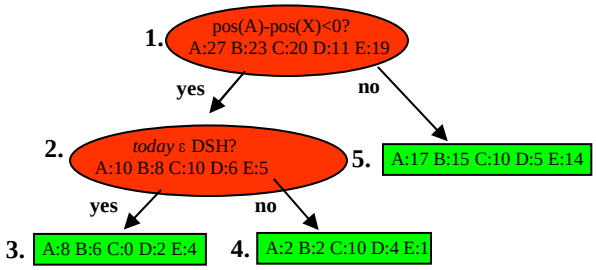


Figure 4. Example of a 5+-choice tree

For "+" DTs, the label closest to the end of the alphabet ("E" in **Figure 4**) stands for a class that can include more than one segment. *E.g.*, if this 5+-choice DT is used to estimate probabilities for a 7-segment RS, the segment closest to X is labeled "A", the second closest "B", the third closest "C", and the fourth closest "D". That leaves 3 segments, all labeled "E". The DT shown yields probability Pr(E) that one of these three will be chosen. Currently, we apply a uniform distribution within this "furthest from X" class, so the probability of any one of the three "E" segments is estimated as Pr(E)/3.

To train the DTs, we generate data items from the second-pass DSH corpus. Each DSH generates several data items. *E.g.*, moving across a seven-segment DSH from left to right, there is an example of the seven-choice case, then one of the six-choice case, *etc*. Thus, this DSH provides three items for training the 5+-choice DT and one item

each for training the 4-choice, 3-choice, and 2-choice DTs. The DT training method was based on Gelfand-Ravishankar-Delp expansion-pruning (Gelfand *et al.*, 1991), for DTs whose nodes contain probability distributions (Lazaridès *et al.*, 1996).

4 Disperp Experiments

We carried out SCM disperp experiments for the English-Chinese task, in both directions. That is, we trained and tested models both for the distortion of English into Chinese-like phrase order, and the distortion of Chinese into English-like phrase order. For reasons of space, details about the "distorted English" experiments won't be given here. Training and development data for the distorted Chinese experiments were taken from the NIST 2005 release of the FBIS corpus of Xinhua news stories. The training corpus comprised 62,000 FBIS segment alignments, and the development "dev" corpus comprised a disjoint set of 2,306 segment alignments from the same FBIS corpus. All disperp results are obtained by testing on "dev" corpus.

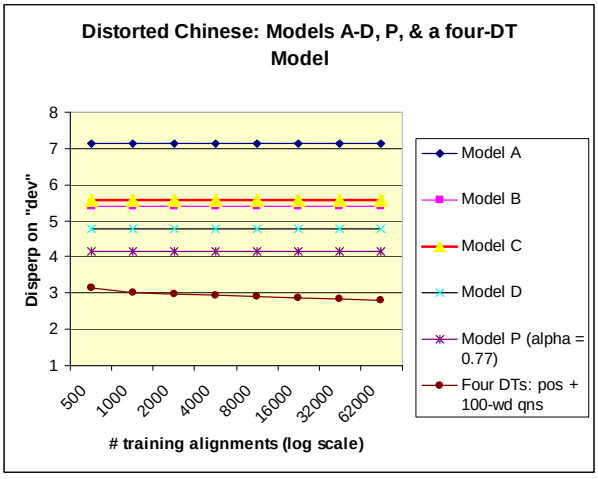


Figure 5. Several SCMs for distorted Chinese

Figure 5 shows disperp results for the models described earlier. The y axis begins at 1.0 (minimum value of disperp). The x axis shows number of alignments (DSHs) used to train DTs, on a log scale. Models A-D are fixed in advance; Model P's single parameter α was optimized once on the entire training set of 62K FBIS alignments (to 0.77) rather than separately for each amount of training data. Model P, the normalized version of Koehn's distortion penalty, is superior to Models A-D, and the DT-based SCM is superior to Model P.

The **Figure 5** DT-based SCM had four trees (2-choice, 3-choice, 4-choice, and 5+-choice) with position-based and word-based questions. The word-based questions involved only the 100 most frequent Chinese words in the training corpus. The system's disperp drops from 3.1 to 2.8 as the number of alignments goes from 500 to 62K.

Figure 6 examines the effect of allowing word-based questions. These questions provide a significant disperp improvement, which grows with the amount of training data.

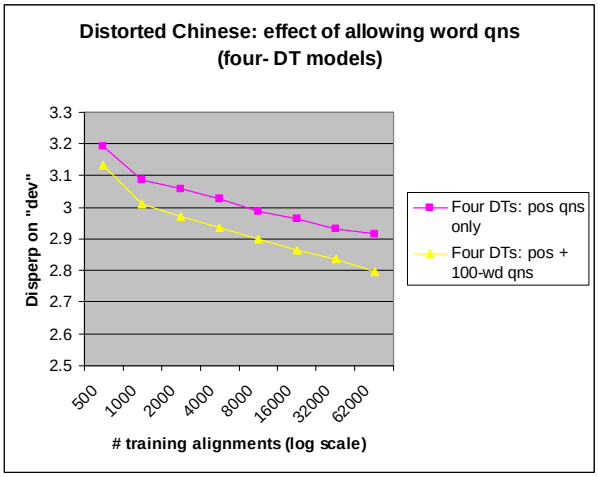


Figure 6. Do word-based questions help?

In the "four-DT" results above, examples with five or more segments are handled by the same "5+-choice" tree. Increasing the number of trees allows finer modeling of multi-segment cases while spreading the training data more thinly. Thus, the optimal number of trees depends on the amount of training data. Fixing this amount to 32K alignments, we varied the number of trees. **Figure 7** shows that this parameter has a significant impact on disperp, and that questions based on the most frequent 100 Chinese words help performance for any number of trees.

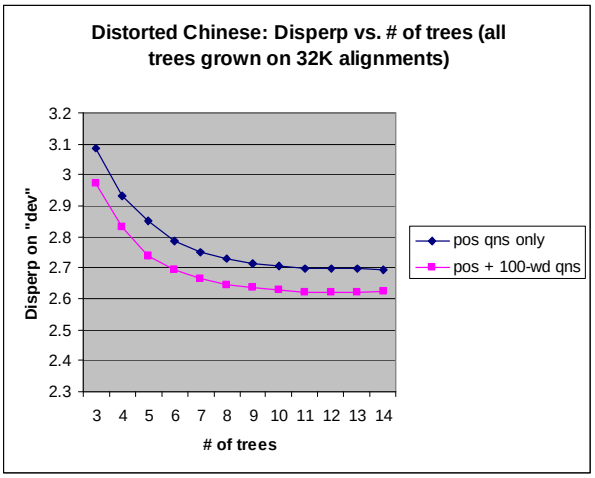


Figure 7. Varying the number of DTs

In **Figure 8** the number of the most frequent Chinese words for questions is varied (for a 13-DT system trained on 32K alignments). Most of the improvement came from the 8 most frequent words, especially from the most frequent, the comma ",". This behaviour seems to be specific to Chinese. In our "distorted English" experiments, questions about the 8 most frequent words also gave a significant improvement, but each of the 8 words had a fairly equal share in the improvement.

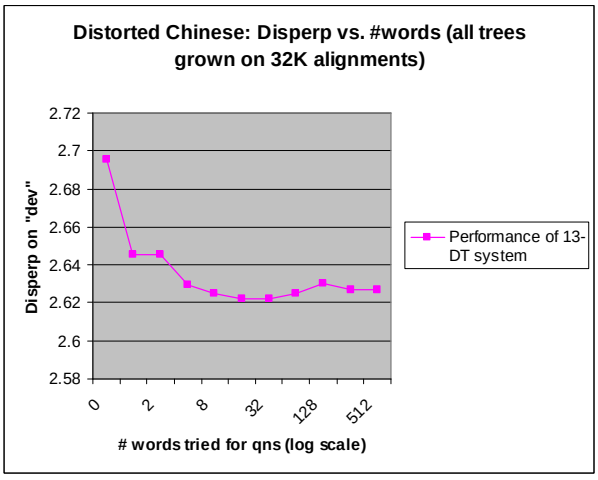


Figure 8. Varying #words (13-DT system)

Finally, we grew the DT system used for the MT experiments: one with 13 trees and questions about the 25 most frequent Chinese words, grown on 88K alignments. Its disperp on the "dev" used for the MT experiments (a different "dev" from the one above – see **Sec. 5.2**) was 2.42 *vs.* 3.48 for the baseline Model P system: a 30% drop.

5 Machine Translation Experiments

5.1 SCMs for Decoding

SCMs assume that the source sentence is fully segmented throughout decoding. Thus, the system must guess the segmentation for the unconsumed part of the source ("remaining source": RS). For the results below, we used a simple heuristic: RS is broken into one-word segments. In future, we will apply a more realistic segmentation model to RS (or modify DT training to reflect accurately RS treatment during decoding).

5.2 Chinese-to-English MT Experiments

The training corpus for the MT system's phrase tables consists of all parallel text available for the NIST MT05 Chinese-English evaluation, except the Xinhua corpora and part 3 of LDC's "Multiple-Translation Chinese Corpus" (MTCCp3). The English language model was trained on the same corpora, plus 250M words from Gigaword. The DT-based SCM was trained and tuned on a subset of this same training corpus (above). The dev corpus for optimizing component weights is MTCCp3. The experimental results below were obtained by testing on the evaluation set for MTeval NIST04.

Phrase tables were learned from the training corpus using the "diag-and" method (Koehn *et al.*, 2003), and using IBM model 2 to produce initial word alignments (these authors found this worked as well as IBM4). Phrase probabilities were based on unsmoothed relative frequencies. The model used by the decoder was a log-linear combination of a phrase translation model (only in the P(source|target) direction), trigram language model, word penalty (lexical weighting), an optional segmentation model (in the form of a phrase penalty) and distortion model. Weights on the components were assigned using the (Och, 2003) method for max-BLEU training on the development set. The decoder uses a dynamic-programming beam-search, like the one in (Koehn, 2004). Future-cost estimates for all distortion models are assigned using the baseline penalty model.

5.3 Decoding Results

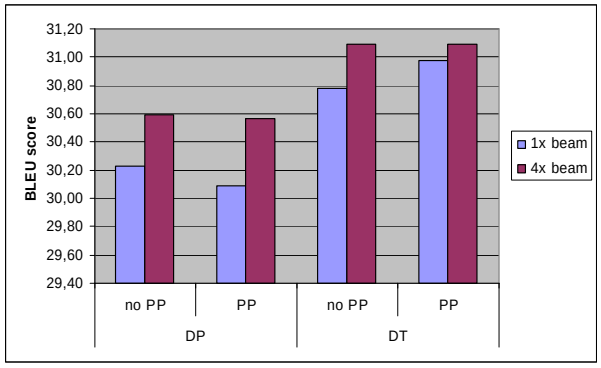


Figure 9. BLEU on NIST04 (95% conf. = ±0.7)

Figure 9 shows experimental results. The "DP" systems use the distortion penalty in (Koehn, 2004) with α optimized on "dev", while "DT" systems use the DT-based SCM. "1x" is the default beam width, while "4x" is a wider beam (our notation reflects decoding time, so "4x" takes four times as long as "1x"). "PP" denotes presence of the phrase penalty component. The advantage of DTs as measured by difference between the score of the best DT system and the best DP system is 0.75 BLEU at 1x and 0.5 BLEU at 4x. With a 95% bootstrap confidence interval of ±0.7 BLEU (based on 1000-fold resampling), the resolution of these results is too coarse to draw firm conclusions.

Thus, we carried out another 1000-fold bootstrap resampling test on NIST04, this time for pairwise system comparison. **Table 1** shows results for BLEU comparisons between the systems with the default (1x) beam. The entries show how often the **A** system (columns) had a better score than the **B** system (rows), in 1000 observations.

A → ***vs.* B ↓**	**DP, no PP**	**DP, PP**	**DT, no PP**	**DT, PP**
DP, no PP	x	2.95%	**99.45%**	**99.55%**
DP, PP	97.05%	x	**99.95%**	**99.95%**
DT, no PP	0.55%	0.05%	x	65.68%
DT, PP	0.45%	0.05%	34.32%	x

Table 1. Pairwise comparison for 1x systems

The table shows that both DT-based 1x systems performed better than either of the DP systems more than 99% of the time (underlined results). Though not shown in the table, the same was true with 4x beam search. The DT 1x system with a phrase penalty had a higher score than the DT 1x system without one about 66% of the time.

6 Summary and Discussion

In this paper, we presented a new class of probabilistic model for distortion, based on the choices made during translation. Unlike some recent distortion models (Kumar and Byrne, 2005; Tillmann and Zhang, 2005; Tillmann, 2004) these ***Segment Choice Models*** (SCMs) allow phrases to be moved globally, between any positions in the sentence. They also lend themselves to quick offline comparison by means of a new metric called ***disperp***. We developed a decision-tree (DT) based SCM whose parameters were optimized on a "dev" corpus via disperp. Two variants of the DT system were experimentally compared with two systems with a distortion penalty on a Chinese-to-English task. In pairwise bootstrap comparisons, the systems with DT-based distortion outperformed the penalty-based systems more than 99% of the time.

The computational cost of training the DTs on large quantities of data is comparable to that of training phrase tables on the same data - large but manageable – and increases linearly with the amount of training data. However, currently there **is** a major problem with DT training: the low proportion of Chinese-English sentence pairs that can be fully segment-aligned and thus be used for DT training (about 27%). This may result in selection bias that impairs performance. We plan to implement an alignment algorithm with smoothed phrase tables (Johnson *et al.* 2006) to achieve segment alignment on 100% of the training data.

Decoding time with the DT-based distortion model is roughly proportional to the square of the number of tokens in the source sentence. Thus, long sentences pose a challenge, particularly during the weight optimization step. In experiments on other language pairs reported elsewhere (Johnson *et al.* 2006), we applied a heuristic: DT training and decoding involved source sentences with 60 or fewer tokens, while longer sentences were handled with the distortion penalty. A more principled ap-

proach would be to divide long source sentences into chunks not exceeding 60 or so tokens, within each of which reordering is allowed, but which cannot themselves be reordered.

The experiments above used a segmentation model that was a count of the number of source segments (sometimes called "phrase penalty"), but we are currently exploring more sophisticated models. Once we have found the best segmentation model, we will improve the system's current naïve single-word segmentation of the remaining source sentence during decoding, and construct a more accurate future cost function for beam search. Another obvious system improvement would be to incorporate more advanced word-based features in the DTs, such as questions about word classes (Tillmann and Zhang 2005, Tillmann 2004).

We also plan to apply SCMs to rescoring N-best lists from the decoder. For rescoring, one could apply several SCMs, some with assumptions differing from those of the decoder. *E.g.*, one could apply right-to-left SCMs, or "distorted target" SCMs which assume a target hypothesis generated the source sentence, instead of vice versa.

Finally, we are contemplating an entirely different approach to DT-based SCMs for decoding. In this approach, only one DT would be used, with only two output classes that could be called "C" and "N". The input to such a tree would be a particular segment in the remaining source sentence, with contextual information (*e.g.*, the sequence of segments already chosen). The DT would estimate the probability Pr(C) that the specified segment is "chosen" and the probability Pr(N) that it is "not chosen". This would eliminate the need to guess the segmentation of the remaining source sentence.

References

P. Brown, S. Della Pietra, V. Della Pietra, and R. Mercer. 1993. "The Mathematics of Statistical Machine Translation: Parameter Estimation". *Computational Linguistics*, 19(2), pp. 263-311.

M. Collins, P. Koehn, and I. Kučerová. 2005. "Clause Restructuring for Statistical Machine Translation". *Proc. ACL*, Ann Arbor, USA, pp. 531-540.

S. Gelfand, C. Ravishankar, and E. Delp. 1991. "An Iterative Growing and Pruning Algorithm for Classification Tree Design". *IEEE Trans. Patt. Analy. Mach. Int.* (*IEEE PAMI*), V. 13, no. 2, pp. 163-174.

F. Jelinek. 1990. "Self-Organized Language Modeling for Speech Recognition" in *Readings in Speech Recognition* (ed. A. Waibel and K. Lee, publ. Morgan Kaufmann), pp. 450-506.

H. Johnson, F. Sadat, G. Foster, R. Kuhn, M. Simard, E. Joanis, and S. Larkin. 2006. "PORTAGE: with Smoothed Phrase Tables and Segment Choice Models". Submitted to *NAACL 2006 Workshop on Statistical Machine Translation*, New York City.

P. Koehn. 2004. "Pharaoh: a Beam Search Decoder for Phrase-Based Statistical Machine Translation Models". *Assoc. Machine Trans. Americas* (*AMTA04*).

P. Koehn, F.-J. Och and D. Marcu. 2003. "Statistical Phrase-Based Translation". *Proc. Human Lang. Tech. Conf. N. Am. Chapt. Assoc. Comp. Ling.* (*NAACL03*), pp. 127-133.

S. Kumar and W. Byrne. 2005. "Local Phrase Reordering Models for Statistical Machine Translation". *HLT/EMNLP*, pp. 161-168, Vancouver, Canada.

A. Lazaridès, Y. Normandin, and R. Kuhn. 1996. "Improving Decision Trees for Acoustic Modeling". *Int. Conf. Spoken Lang. Proc.* (*ICSLP96*), V. 2, pp. 1053-1056, Philadelphia, Pennsylvania, USA.

F. Och and H. Ney. 2004. "The Alignment Template Approach to Statistical Machine Translation". *Comp. Linguistics*, V. 30, Issue 4, pp. 417-449.

Franz Josef Och. 2003. "Minimum Error Rate Training for Statistical Machine Translation". *Proc. ACL*, Sapporo, Japan.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. "BLEU: A method for automatic evaluation of machine translation". *Proc. ACL*, pp. 311-318.

C. Tillmann and T. Zhang. 2005. "A Localized Prediction Model for Statistical Machine Translation". *Proc. ACL*.

C. Tillmann. 2004. "A Block Orientation Model for Statistical Machine Translation". *HLT/NAACL*.

S. Vogel, H. Ney, and C. Tillmann. 1996. "HMM-Based Word Alignment in Statistical Translation". *COLING*, pp. 836-841.

Effectively Using Syntax for Recognizing False Entailment

Rion Snow
Computer Science Department
Stanford University
Stanford, CA 94305
rion@cs.stanford.edu

Lucy Vanderwende and Arul Menezes
Microsoft Research
One Microsoft Way
Redmond, WA 98027
{lucyv,arulm}@microsoft.com

Abstract

Recognizing textual entailment is a challenging problem and a fundamental component of many applications in natural language processing. We present a novel framework for recognizing textual entailment that focuses on the use of syntactic heuristics to recognize false entailment. We give a thorough analysis of our system, which demonstrates state-of-the-art performance on a widely-used test set.

1 Introduction

Recognizing the semantic equivalence of two fragments of text is a fundamental component of many applications in natural language processing. Recognizing textual entailment, as formulated in the recent PASCAL Challenge [1], is the problem of determining whether some *text sentence* T entails some *hypothesis sentence* H.

The motivation for this formulation was to isolate and evaluate the application-independent component of semantic inference shared across many application areas, reflected in the division of the PASCAL RTE dataset into seven distinct tasks: Information Extraction (IE), Comparable Documents (CD), Reading Comprehension (RC), Machine Translation (MT), Information Retrieval (IR), Question Answering (QA), and Paraphrase Acquisition (PP).

[1] http://www.pascal-network.org/Challenges/RTE. The examples given throughout this paper are from the first PASCAL RTE dataset, described in Section 6.

The RTE problem as presented in the PASCAL RTE dataset is particularly attractive in that it is a reasonably simple task for human annotators with high inter-annotator agreement (95.1% in one independent labeling (Bos and Markert, 2005)), but an extremely challenging task for automated systems. The highest accuracy systems on the RTE test set are still much closer in performance to a random baseline accuracy of 50% than to the inter-annotator agreement. For example, two high-accuracy systems are those described in (Tatu and Moldovan, 2005), achieving 60.4% accuracy with no task-specific information, and (Bos and Markert, 2005), which achieves 61.2% *task-dependent* accuracy, i.e. when able to use the specific task labels as input.

Previous systems for RTE have attempted a wide variety of strategies. Many previous approaches have used a logical form representation of the text and hypothesis sentences, focusing on deriving a proof by which one can infer the hypothesis logical form from the text logical form (Bayer et al., 2005; Bos and Markert, 2005; Raina et al., 2005; Tatu and Moldovan, 2005). These papers often cite that a major obstacle to accurate theorem proving for the task of textual entailment is the lack of world knowledge, which is frequently difficult and costly to obtain and encode. Attempts have been made to remedy this deficit through various techniques, including model-building (Bos and Markert, 2005) and the addition of semantic axioms (Tatu and Moldovan, 2005).

Our system diverges from previous approaches most strongly by focusing upon false entailments; rather than assuming that a given entailment is false until proven true, we make the opposite assump-

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 33–40,
New York, June 2006.

tion, and instead focus on applying knowledge-free heuristics that can act locally on a subgraph of syntactic dependencies to determine with high confidence that the entailment is false. Our approach is inspired by an analysis of the RTE dataset that suggested a syntax-based approach should be approximately twice as effective at predicting false entailment as true entailment (Vanderwende and Dolan, 2006). The analysis implied that a great deal of syntactic information remained unexploited by existing systems, but gave few explicit suggestions on how syntactic information should be applied; this paper provides a starting point for creating the heuristics capable of obtaining the bound they suggest[2].

2 System Description

Similar to most other syntax-based approaches to recognizing textual entailment, we begin by representing each text and hypothesis sentence pair in *logical forms*. These logical forms are generated using NLPWIN[3], a robust system for natural language parsing and generation (Heidorn, 2000). Our logical form representation may be considered equivalently as a set of triples of the form RELATION$(node_i, node_j)$, or as a graph of syntactic dependencies; we use both terminologies interchangeably. Our algorithm proceeds as follows:

1. Parse each sentence with the NLPWIN parser, resulting in syntactic dependency graphs for the text and hypothesis sentences.
2. Attempt an alignment of each *content* node in the dependency graph of the hypothesis sentence to some node in the graph of the text sentence, using a set of heuristics for alignment (described in Section 3).
3. Using the alignment, apply a set of syntactic heuristics for recognizing false entailment (described in Section 4); if any match, predict that the entailment is false.

[2](Vanderwende and Dolan, 2006) suggest that the truth or falsehood of 48% of the entailment examples in the RTE test set could be correctly identified via syntax and a thesaurus alone; thus by random guessing on the rest of the examples one might hope for an accuracy level of $0.48 + \frac{0.52}{2} = 74\%$.

[3]To aid in the replicability of our experiments, we have published the NLPWIN logical forms for all sentences from the development and test sets in the PASCAL RTE dataset at http://research.microsoft.com/nlp/Projects/RTE.aspx.

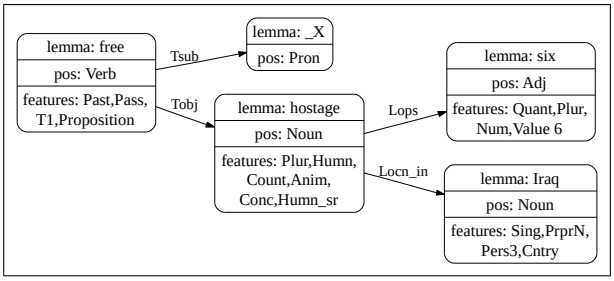


Figure 1: Logical form produced by NLPWIN for the sentence "Six hostages in Iraq were freed."

4. If no syntactic heuristic matches, back off to a lexical similarity model (described in section 5.1), with an attempt to align detected paraphrases (described in section 5.2).

In addition to the typical syntactic information provided by a dependency parser, the NLPWIN parser provides an extensive number of semantic features obtained from various linguistic resources, creating a rich environment for feature engineering. For example, Figure 1 (from Dev Ex. #616) illustrates the dependency graph representation we use, demonstrating the stemming, part-of-speech tagging, syntactic relationship identification, and semantic feature tagging capabilities of NLPWIN.

We define a *content* node to be any node whose lemma is not on a small stoplist of common stop words. In addition to content vs. non-content nodes, among content nodes we distinguish between *entities* and *nonentities*: an *entity* node is any node classified by the NLPWIN parser as being a proper noun, quantity, or time.

Each of the features of our system were developed from inspection of sentence pairs from the RTE development data set, and used in the final system only if they improved the system's accuracy on the development set (or improved F-score if accuracy was unchanged); sentence pairs in the RTE test set were left uninspected and used for testing purposes only.

3 Linguistic cues for node alignment

Our syntactic heuristics for recognizing false entailment rely heavily on the correct alignment of words and multiword units between the text and hypothesis logical forms. In the notation below, we will consider h and t to be nodes in the hypothesis H and

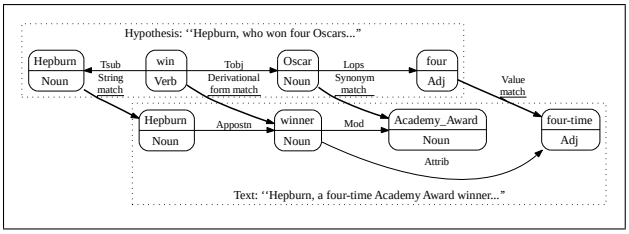


Figure 2: Example of synonym, value, and derivational form alignment heuristics, Dev Ex. #767

text T logical forms, respectively. To accomplish the task of node alignment we rely on the following heuristics:

3.1 WordNet synonym match

As in (Herrera et al., 2005) and others, we align a node $h \in H$ to any node $t \in T$ that has both the same part of speech and belongs to the same synset in WordNet. Our alignment considers multiword units, including compound nouns (e.g., we align "Oscar" to "Academy Award" as in Figure 2), as well as verb-particle constructions such as "set off" (aligned to "trigger" in Test Ex. #1983).

3.2 Numeric value match

The NLPWIN parser assigns a normalized numeric value feature to each piece of text inferred to correspond to a numeric value; this allows us to align "6th" to "sixth" in Test Ex. #1175. and to align "a dozen" to "twelve" in Test Ex. #1231.

3.3 Acronym match

Many acronyms are recognized using the synonym match described above; nonetheless, many acronyms are not yet in WordNet. For these cases we have a specialized acronym match heuristic which aligns pairs of nodes with the following properties: if the lemma for some node h consists only of capitalized letters (with possible interceding periods), and the letters correspond to the first characters of some multiword lemma for some $t \in T$, then we consider h and t to be aligned. This heuristic allows us to align "UNDP" to "United Nations Development Programme" in Dev Ex. #357 and "ANC" to "African National Congress" in Test Ex. #1300.

3.4 Derivational form match

We would like to align words which have the same root form (or have a synonym with the same root form) and which possess similar semantic meaning, but which may belong to different syntactic categories. We perform this by using a combination of the synonym and derivationally-related form information contained within WordNet. Explicitly our procedure for constructing the set of derivationally-related forms for a node h is to take the union of all derivationally-related forms of all the synonyms of h (including h itself), i.e.:

$$\text{DERIV}(h) = \cup_{s \in \text{WN-SYN}(h)} \text{WN-DERIV}(s)$$

In addition to the noun/verb derivationally-related forms, we detect adjective/adverb derivationally-related forms that differ only by the suffix 'ly'.

Unlike the previous alignment heuristics, we do not expect that two nodes aligned via derivationally-related forms will play the same syntactic role in their respective sentences. Thus we consider two nodes aligned in this way to be *soft-aligned*, and we do not attempt to apply our false entailment recognition heuristics to nodes aligned in this way.

3.5 Country adjectival form / demonym match

As a special case of derivational form match, we soft-align matches from an explicit list of place names, adjectival forms, and demonyms[4]; e.g., "Sweden" and "Swedish" in Test Ex. #1576.

3.6 Other heuristics for alignment

In addition to these heuristics, we implemented a hyponym match heuristic similar to that discussed in (Herrera et al., 2005), and a heuristic based on the string-edit distance of two lemmas; however, these heuristics yielded a decrease in our system's accuracy on the development set and were thus left out of our final system.

4 Recognizing false entailment

The bulk of our system focuses on heuristics for recognizing false entailment. For purposes of notation, we define binary functions for the existence

[4]List of adjectival forms and demonyms based on the list at: http://en.wikipedia.org/wiki/List_of_demonyms

Unaligned Entity:	$\text{ENTITY}(h) \wedge \forall t. \neg \text{ALIGN}(h,t) \rightarrow False.$
Negation Mismatch:	$\text{ALIGN}(h,t) \wedge \text{NEG}(t) \neq \text{NEG}(h) \rightarrow False.$
Modal Mismatch:	$\text{ALIGN}(h,t) \wedge \text{MOD}(t) \wedge \neg\text{MOD}(h) \rightarrow False.$
Antonym Match:	$\text{ALIGN}(h_1,t_1) \wedge \text{REL}(h_0,h_1) \wedge \text{REL}(t_0,t_1) \wedge \text{LEMMA}(t_0) \in \text{ANTONYMS}(h_0) \rightarrow False$
Argument Movement:	$\text{ALIGN}(h_1,t_1) \wedge \text{ALIGN}(h_2,t_2) \wedge \text{REL}(h_1,h_2) \wedge \neg\text{REL}(t_1,t_2) \wedge \text{REL} \in \{\text{SUBJ, OBJ, IND}\} \rightarrow False$
Superlative Mismatch:	$\neg(\text{SUPR}(h_1) \rightarrow (\text{ALIGN}(h_1,t_1) \wedge \text{ALIGN}(h_2,t_2) \wedge \text{REL}_1(h_2,h_1) \wedge \text{REL}_1(t_2,t_1)$ $\wedge\forall t_3.(\text{REL}_2(t_2,t_3) \wedge \text{REL}_2 \in \{\text{MOD,POSSR,LOCN}\} \rightarrow \text{REL}_2(h_2,h_3) \wedge \text{ALIGN}(h_3,t_3))) \rightarrow False$
Conditional Mismatch:	$\text{ALIGN}(h_1,t_1) \wedge \text{ALIGN}(h_2,t_2) \wedge \text{COND} \in \text{PATH}(t_1,t_2) \wedge \text{COND} \notin \text{PATH}(h_1,h_2) \rightarrow False$

Table 1: Summary of heuristics for recognizing false entailment

of each semantic node feature recognized by NLPWIN; e.g., if h is negated, we state that $\text{NEG}(h) = \text{TRUE}$. Similarly we assign binary functions for the existence of each syntactic relation defined over pairs of nodes. Finally, we define the function $\text{ALIGN}(h,t)$ to be true if and only if the node $h \in H$ has been 'hard-aligned' to the node $t \in T$ using one of the heuristics in Section 3. Other notation is defined in the text as it is used. Table 1 summarizes all heuristics used in our final system to recognize false entailment.

4.1 Unaligned entity

If some node h has been recognized as an entity (i.e., as a proper noun, quantity, or time) but has not been aligned to any node t, we predict that the entailment is false. For example, we predict that Test Ex. #1863 is false because the entities "Suwariya", "20 miles", and "35" in H are unaligned.

4.2 Negation mismatch

If any two nodes (h,t) are aligned, and one (and only one) of them is negated, we predict that the entailment is false. Negation is conveyed by the NEG feature in NLPWIN. This heuristic allows us to predict false entailment in the example "Pertussis is not very contagious" and "...pertussis, is a highly contagious bacterial infection" in Test Ex. #1144.

4.3 Modal auxiliary verb mismatch

If any two nodes (h,t) are aligned, and t is modified by a modal auxiliary verb (e.g, *can*, *might*, *should*, etc.) but h is not similarly modified, we predict that the entailment is false. Modification by a modal auxiliary verb is conveyed by the MOD feature in NLPWIN. This heuristic allows us to predict false entailment between the text phrase "would constitute a threat to democracy", and the hypothesis phrase "constitutes a democratic threat" in Test Ex. #1203.

4.4 Antonym match

If two aligned noun nodes (h_1,t_1) are both subjects or both objects of verb nodes (h_0,t_0) in their respective sentences, i.e., $\text{REL}(h_0,h_1) \wedge \text{REL}(t_0,t_1) \wedge \text{REL} \in \{\text{SUBJ,OBJ}\}$, then we check for a verb antonym match between (h_0,t_0). We construct the set of verb antonyms using WordNet; we consider the antonyms of h_0 to be the union of the antonyms of the first three senses of $\text{LEMMA}(h_0)$, or of the nearest antonym-possessing hypernyms if those senses do not themselves have antonyms in WordNet. Explicitly our procedure for constructing the antonym set of a node h_0 is as follows:

1. $\text{ANTONYMS}(h_0) = \{\}$
2. For each of the first three listed senses s of $\text{LEMMA}(h_0)$ in WordNet:
   (a) While $|\text{WN-ANTONYMS}(s)| = 0$
      i. $s \leftarrow \text{WN-HYPERNYM}(s)$
   (b) $\text{ANTONYMS}(h_0) \leftarrow \text{ANTONYMS}(h_0) \cup \text{WN-ANTONYMS}(s)$
3. return $\text{ANTONYMS}(h_0)$

In addition to the verb antonyms in WordNet, we detect the prepositional antonym pairs (*before/after*, *to/from*, and *over/under*). This heuristic allows us to predict false entailment between "Black holes can lose mass..." and "Black holes can regain some of their mass..." in Test Ex. #1445.

4.5 Argument movement

For any two aligned verb nodes (h_1,t_1), we consider each noun child h_2 of h_1 possessing any of

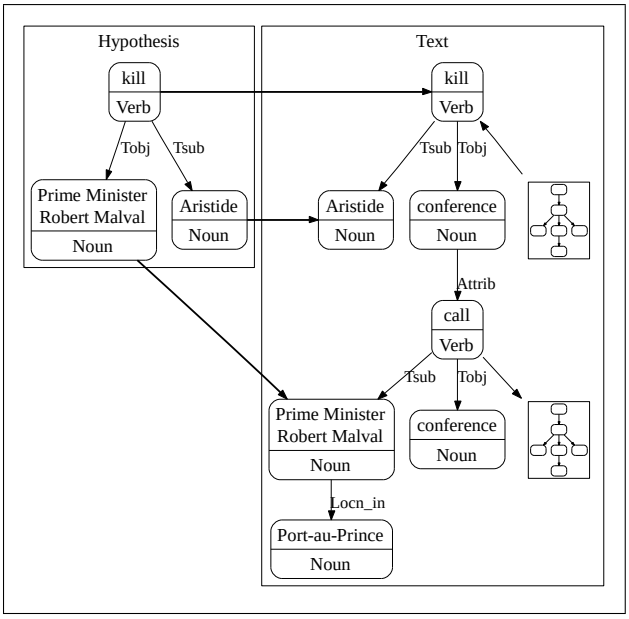


Figure 3: Example of object movement signaling false entailment

the subject, object, or indirect object relations to h_1, i.e., there exists $\text{REL}(h_1, h_2)$ such that $\text{REL} \in \{\text{SUBJ}, \text{OBJ}, \text{IND}\}$. If there is some node t_2 such that $\text{ALIGN}(h_2, t_2)$, but $\text{REL}(t_1, t_2) \neq \text{REL}(h_1, h_2)$, then we predict that the entailment is false.

As an example, consider Figure 3, representing subgraphs from Dev Ex. #1916:

T: ...U.N. officials are also dismayed that Aristide killed a conference called by Prime Minister Robert Malval...

H: Aristide kills Prime Minister Robert Malval.

Here let (h_1, t_1) correspond to the aligned verbs with lemma *kill*, where the object of h_1 has lemma *Prime Minister Robert Malval*, and the object of t_1 has lemma *conference*. Since h_2 is aligned to some node t_2 in the text graph, but $\neg\text{OBJ}(t_1, t_2)$, the sentence pair is rejected as a false entailment.

4.6 Superlative mismatch

If some adjective node h_1 in the hypothesis is identified as a superlative, check that all of the following conditions are satisfied:

1. h_1 is aligned to some superlative t_1 in the text sentence.

2. The noun phrase h_2 modified by h_1 is aligned to the noun phrase t_2 modified by t_1.

3. Any additional modifier t_3 of the noun phrase t_2 is aligned to some modifier h_3 of h_2 in the hypothesis sentence (reverse subset match).

If any of these conditions are not satisfied, we predict that the entailment is false. This heuristic allows us to predict false entailment in (Dev Ex. #908):

T: Time Warner is the world's largest media and Internet company.

H: Time Warner is the world's largest company.

Here "largest media and Internet company" in T fails the reverse subset match (condition 3) to "largest company" in H.

4.7 Conditional mismatch

For any pair of aligned nodes (h_1, t_1), if there exists a second pair of aligned nodes (h_2, t_2) such that the shortest path $\text{PATH}(t_1, t_2)$ in the dependency graph T contains the conditional relation, then $\text{PATH}(h_1, h_2)$ must also contain the conditional relation, or else we predict that the entailment is false. For example, consider the following false entailment (Dev Ex. #60):

T: If a Mexican approaches the border, he's assumed to be trying to illegally cross.

H: Mexicans continue to illegally cross border.

Here, "Mexican" and "cross" are aligned, and the path between them in the text contains the conditional relation, but does not in the hypothesis; thus the entailment is predicted to be false.

4.8 Other heuristics for false entailment

In addition to these heuristics, we additionally implemented an IS-A mismatch heuristic, which attempted to discover when an IS-A relation in the hypothesis sentence was not implied by a corresponding IS-A relation in the text; however, this heuristic yielded a loss in accuracy on the development set and was therefore not included in our final system.

5 Lexical similarity and paraphrase detection

5.1 Lexical similarity using MindNet

In case none of the preceding heuristics for rejection are applicable, we back off to a lexical similarity model similar to that described in (Glickman et al., 2005). For every content node $h \in H$

not already aligned by one of the heuristics in Section 3, we obtain a similarity score $\text{MN}(h,t)$ from a similarity database that is constructed automatically from the data contained in MindNet[5] as described in (Richardson, 1997). Our similarity function is thus:

$$sim(h,t) = \begin{cases} 1 & \text{if ANY-ALIGN}(h,t) \\ \text{MN}(h,t) & \text{if MN}(h,t) > min \\ min & \text{otherwise} \end{cases}$$

Where the minimum score min is a parameter tuned for maximum accuracy on the development set; $min = 0.00002$ in our final system. We then compute the entailment score:

$$score(H,T) = \frac{1}{|H|} \prod_{h \in H} \max_{t \in T} sim(h,t)$$

This approach is identical to that used in (Glickman et al., 2005), except that we use alignment heuristics and MindNet similarity scores in place of their web-based estimation of lexical entailment probabilities, and we take as our score the geometric mean of the component entailment scores rather than the unnormalized product of probabilities.

5.2 Measuring phrasal similarity using the web

The methods discussed so far for alignment are limited to aligning pairs of single words or multiple-word units constituting single syntactic categories; these are insufficient for the problem of detecting more complicated paraphrases. For example, consider the following true entailment (Dev Ex. #496):

T: ...Muslims believe there is only one God.

H: Muslims are monotheistic.

Here we would like to align the hypothesis phrase "are monotheistic" to the text phrase "believe there is only one God"; unfortunately, single-node alignment aligns only the nodes with lemma "Muslim". In this section we describe the approach used in our system to approximate phrasal similarity via distributional information obtained using the MSN Search search engine.

We propose a metric for measuring phrasal similarity based on a phrasal version of the distributional hypothesis: we propose that a phrase template P_h (e.g. 'x_h are monotheistic') has high semantic similarity to a template P_t (e.g. "x_t believe there is only one God"), with possible "slot-fillers" x_h and x_t, respectively, if the overlap of the sets of observed slot-fillers $X_h \cap X_t$ for those phrase templates is high in some sufficiently large corpus (e.g., the Web).

To measure phrasal similarity we issue the surface text form of each candidate phrase template as a query to a web-based search engine, and parse the returned sentences in which the candidate phrase occurs to determine the appropriate slot-fillers. For example, in the above example, we observe the set of slot-fillers $X_t = \{$Muslims, Christians, Jews, Saivities, Sikhs, Caodaists, People$\}$, and $X_h \cap X_t = \{$Muslims, Christians, Jews, Sikhs, People$\}$.

Explicitly, given the text and hypothesis logical forms, our algorithm proceeds as follows to compute the phrasal similarity between all phrase templates in H and T:

1. For each pair of aligned single node and unaligned leaf node (t_1, t_l) (or pair of aligned nodes (t_1, t_2)) in the text T:
   (a) Use NLPWIN to generate a surface text string S from the underlying logical form $\text{PATH}(t_1, t_2)$.
   (b) Create the surface string template phrase P_t by removing from S the lemmas corresponding to t_1 (and t_2, if path is between aligned nodes).
   (c) Perform a web search for the string P_t.
   (d) Parse the resulting sentences containing P_t and extract all non-pronoun slot fillers $x_t \in X_t$ that satisfy the same syntactic roles as t_1 in the original sentence.
2. Similarly, extract the slot fillers X_h for each discovered phrase template P_h in H.
3. Calculate paraphrase similarity as a function of the overlap between the slot-filler sets X_t and X_h, i.e: $score(P_h, P_t) = \frac{|X_h \cap X_t|}{|X_t|}$.

We then incorporate paraphrase similarity within the lexical similarity model by allowing, for some unaligned node $h \in P_h$, where $t \in P_t$:

$$sim(h,t) = \max(\text{MN}(h,t), score(P_h, P_t))$$

[5] http://research.microsoft.com/mnex

Our approach to paraphrase detection is most similar to the TE/ASE algorithm (Szpektor et al., 2004), and bears similarity to both DIRT (Lin and Pantel, 2001) and KnowItAll (Etzioni et al., 2004). The chief difference in our algorithm is that we generate the surface text search strings from the parsed logical forms using the generation capabilities of NLPWIN (Aikawa et al., 2001), and we verify that the syntactic relations in each discovered web snippet are isomorphic to those in the original candidate paraphrase template.

6 Results and Discussion

In this section we present the final results of our system on the PASCAL RTE-1 test set, and examine our features in an ablation study. The PASCAL RTE-1 development and test sets consist of 567 and 800 examples, respectively, with the test set split equally between true and false examples.

6.1 Results and Performance Comparison on the PASCAL RTE-1 Test Set

Table 2 displays the accuracy and confidence-weighted score[6] (CWS) of our final system on each of the tasks for both the development and test sets.

Our overall test set accuracy of 62.50% represents a 2.1% absolute improvement over the task-independent system described in (Tatu and Moldovan, 2005), and a 20.2% relative improvement in accuracy over their system with respect to an uninformed baseline accuracy of 50%.

To compute confidence scores for our judgments, any entailment determined to be false by any heuristic was assigned maximum confidence; no attempts were made to distinguish between entailments rejected by different heuristics. The confidence of all other predictions was calculated as the absolute value in the difference between the output $score(H, T)$ of the lexical similarity model and the threshold $t = 0.1285$ as tuned for highest accuracy on our development set. We would expect a higher CWS to result from learning a more appropriate confidence function; nonetheless our overall test set CWS of 0.6534 is higher than previously-reported task-independent systems (however, the task-dependent system reported in (Raina et al., 2005) achieves a CWS of 0.686).

[6] As in (Dagan et al., 2005) we compute the confidence-weighted score (or "average precision") over n examples $\{c_1, c_2, ..., c_n\}$ ranked in order of decreasing confidence as $cws = \frac{1}{n} \sum_{i=1}^{n} \frac{(\#\text{correct-up-to-rank-}i)}{i}$

	Dev Set		Test Set	
Task	**acc**	**cws**	**acc**	**cws**
CD	0.8061	0.8357	0.7867	0.8261
RC	0.5534	0.5885	0.6429	0.6476
IR	0.6857	0.6954	0.6000	0.6571
MT	0.7037	0.7145	0.6000	0.6350
IE	0.5857	0.6008	0.5917	0.6275
QA	0.7111	0.7121	0.5308	0.5463
PP	0.7683	0.7470	0.5200	0.5333
All	0.6878	0.6888	0.6250	0.6534

Table 2: Summary of accuracies and confidence-weighted scores, by task

Alignment Feature	**Dev**	**Test**
Synonym Match	0.0106	0.0038
Derivational Form	0.0053	0.0025
Paraphrase	0.0053	0.0000
Lexical Similarity	0.0053	0.0000
Value Match	0.0017	0.0013
Acronym Match	0.0017	0.0013
Adjectival Form[7]	0.0000	0.0063
False Entailment Feature	**Dev**	**Test**
Negation Mismatch	0.0106	0.0025
Argument Movement	0.0070	0.0250
Conditional Mismatch	0.0053	0.0037
Modal Mismatch	0.0035	0.0013
Superlative Mismatch	0.0035	-0.0025
Entity Mismatch	0.0018	0.0063

Table 3: Feature ablation study; quantity is the accuracy loss obtained by removal of single feature

6.2 Feature analysis

Table 3 displays the results of our feature ablation study, analyzing the individual effect of each feature.

Of the seven heuristics used in our final system for node alignment (including lexical similarity and paraphrase detection), our ablation study showed

[7] As discussed in Section 2, features with no effect on development set accuracy were included in the system if and only if they improved the system's unweighted F-score.

that five were helpful in varying degrees on our test set, but that removal of either MindNet similarity scores or paraphrase detection resulted in no accuracy loss on the test set.

Of the six false entailment heuristics used in the final system, five resulted in an accuracy improvement on the test set (the most effective by far was the "Argument Movement", resulting in a net gain of 20 correctly-classified false examples); inclusion of the "Superlative Mismatch" feature resulted in a small net loss of two examples.

We note that our heuristics for false entailment, where applicable, were indeed significantly more accurate than our final system as a whole; on the set of examples predicted false by our heuristics we had 71.3% accuracy on the training set (112 correct out of 157 predicted), and 72.9% accuracy on the test set (164 correct out of 225 predicted).

7 Conclusion

In this paper we have presented and analyzed a system for recognizing textual entailment focused primarily on the recognition of *false* entailment, and demonstrated higher performance than achieved by previous approaches on the widely-used PASCAL RTE test set. Our system achieves state-of-the-art performance despite not exploiting a wide array of sources of knowledge used by other high-performance systems; we submit that the performance of our system demonstrates the unexploited potential in features designed specifically for the recognition of false entailment.

Acknowledgments

We thank Chris Brockett, Michael Gamon, Gary Kacmarick, and Chris Quirk for helpful discussion. Also, thanks to Robert Ragno for assistance with the MSN Search API. Rion Snow is supported by an NDSEG Fellowship sponsored by the DOD and AFOSR.

References

Takako Aikawa, Maite Melero, Lee Schwartz, and Andi Wu. 2001. Multilingual Sentence Generation. In *Proc. of 8^{th} European Workshop on Natural Language Generation*.

Samuel Bayer, John Burger, Lisa Ferro, John Henderson, and Alexander Yeh. 2005. MITRE's Submissions to the EU Pascal RTE Challenge. In *Proc. of the PASCAL Challenges Workshop on RTE 2005*.

Johan Bos and Katja Markert. 2005. Recognizing Textual Entailment with Logical Inference. In *Proc. HLT-EMNLP 2005*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the PASCAL Challenges Workshop on RTE 2005*.

Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-scale information extraction in KnowItAll. In *Proc. WWW 2004*.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Mass.

Oren Glickman, Ido Dagan, and Moshe Koppel. 2005. Web Based Probabilistic Textual Entailment. In *Proc. of the PASCAL Challenges Workshop on RTE 2005*.

George E. Heidorn. 2000. Intelligent Writing Assistance. In R. Dale, H. Moisl, and H. Somers (eds.), *A Handbook of Natural Language Processing: Techniques and Applications for the Processing of Language as Text*. Marcel Dekker, New York. 181-207.

Jesús Herrera, Anselmo Peñas, and Felisa Verdejo. 2005. Textual Entailment Recognision Based on Dependency Analysis and WordNet. In *Proc. of the PASCAL Challenges Workshop on RTE 2005*.

Dekang Lin and Patrick Pantel. 2001. DIRT - Discovery of Inference Rules from Text. In *Proc. KDD 2001*.

Rajat Raina, Andrew Y. Ng, and Christopher D. Manning. 2005. Robust textual inference via learning and abductive reasoning. In *Proc. AAAI 2005*.

Stephen D. Richardson. 1997. Determining Similarity and Inferring Relations in a Lexical Knowledge Base. Ph.D. thesis, The City University of New York.

Idan Szpektor, Hristo Tanev, Ido Dagan, and Bonaventura Coppola. 2004. Scaling Web-based Acquisition of Entailment Relations. In *Proc. EMNLP 2004*.

Marta Tatu and Dan Moldovan. 2005. A Semantic Approach to Recognizing Textual Entailment. In *Proc. HLT-EMNLP 2005*.

Lucy Vanderwende and William B. Dolan. 2006. What Syntax Can Contribute in the Entailment Task. In *MLCW 2005*, LNAI 3944, pp. 205–216. J. Quinonero-Candela et al. (eds.). Springer-Verlag.

Learning to recognize features of valid textual entailments

Bill MacCartney, Trond Grenager, Marie-Catherine de Marneffe,
Daniel Cer, and Christopher D. Manning
Computer Science Department
Stanford University
Stanford, CA 94305
{wcmac, grenager, mcdm, cerd, manning}@cs.stanford.edu

Abstract

This paper advocates a new architecture for textual inference in which finding a good alignment is separated from evaluating entailment. Current approaches to semantic inference in question answering and textual entailment have approximated the entailment problem as that of computing the best alignment of the hypothesis to the text, using a locally decomposable matching score. We argue that there are significant weaknesses in this approach, including flawed assumptions of monotonicity and locality. Instead we propose a pipelined approach where alignment is followed by a classification step, in which we extract features representing high-level characteristics of the entailment problem, and pass the resulting feature vector to a statistical classifier trained on development data. We report results on data from the 2005 Pascal RTE Challenge which surpass previously reported results for alignment-based systems.

1 Introduction

During the last five years there has been a surge in work which aims to provide robust textual inference in arbitrary domains about which the system has no expertise. The best-known such work has occurred within the field of question answering (Pasca and Harabagiu, 2001; Moldovan et al., 2003); more recently, such work has continued with greater focus in addressing the PASCAL Recognizing Textual Entailment (RTE) Challenge (Dagan et al., 2005) and within the U.S. Government AQUAINT program. Substantive progress on this task is key to many text and natural language applications. If one could tell that *Protestors chanted slogans opposing a free trade agreement* was a match for *people demonstrating against free trade*, then one could offer a form of semantic search not available with current keyword-based search. Even greater benefits would flow to richer and more semantically complex NLP tasks.

Because full, accurate, open-domain natural language understanding lies far beyond current capabilities, nearly all efforts in this area have sought to extract the maximum mileage from quite limited semantic representations. Some have used simple measures of semantic overlap, but the more interesting work has largely converged on a graph-alignment approach, operating on semantic graphs derived from syntactic dependency parses, and using a locally-decomposable alignment score as a proxy for strength of entailment. (Below, we argue that even approaches relying on weighted abduction may be seen in this light.) In this paper, we highlight the fundamental semantic limitations of this type of approach, and advocate a multi-stage architecture that addresses these limitations. The three key limitations are an *assumption of monotonicity*, an *assumption of locality*, and a *confounding of alignment and evaluation of entailment*.

We focus on the PASCAL RTE data, examples from which are shown in table 1. This data set contains pairs consisting of a short text followed by a one-sentence hypothesis. The goal is to say whether the hypothesis follows from the text and general background knowledge, according to the intuitions of an intelligent human reader. That is, the standard is not whether the hypothesis is logically entailed, but whether it can reasonably be inferred.

2 Approaching a robust semantics

In this section we try to give a unifying overview to current work on robust textual inference, to present fundamental limitations of current methods, and then to outline our approach to resolving them. Nearly all current textual inference systems use a single-stage matching/proof process, and differ

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 41–48,
New York, June 2006.

ID	Text	Hypothesis	Entailed
59	Two Turkish engineers and an Afghan translator kidnapped in December were freed Friday.	translator kidnapped in Iraq	no
98	Sharon warns Arafat could be targeted for assassination.	prime minister targeted for assassination	no
152	Twenty-five of the dead were members of the law enforcement agencies and the rest of the 67 were civilians.	25 of the dead were civilians.	no
231	The memorandum noted the United Nations estimated that 2.5 million to 3.5 million people died of AIDS last year.	Over 2 million people died of AIDS last year.	yes
971	Mitsubishi Motors Corp.'s new vehicle sales in the US fell 46 percent in June.	Mitsubishi sales rose 46 percent.	no
1806	Vanunu, 49, was abducted by Israeli agents and convicted of treason in 1986 after discussing his work as a mid-level Dimona technician with Britain's Sunday Times newspaper.	Vanunu's disclosures in 1968 led experts to conclude that Israel has a stockpile of nuclear warheads.	no
2081	The main race track in Qatar is located in Shahaniya, on the Dukhan Road.	Qatar is located in Shahaniya.	no

Table 1: Illustrative examples from the PASCAL RTE data set, available at *http://www.pascal-network.org/Challenges/RTE*. Though most problems shown have answer *no*, the data set is actually balanced between *yes* and *no*.

mainly in the sophistication of the matching stage. The simplest approach is to base the entailment prediction on the degree of semantic overlap between the text and hypothesis using models based on bags of words, bags of n-grams, TF-IDF scores, or something similar (Jijkoun and de Rijke, 2005). Such models have serious limitations: semantic overlap is typically a symmetric relation, whereas entailment is clearly not, and, because overlap models do not account for syntactic or semantic structure, they are easily fooled by examples like ID 2081.

A more structured approach is to formulate the entailment prediction as a graph matching problem (Haghighi et al., 2005; de Salvo Braz et al., 2005). In this formulation, sentences are represented as normalized syntactic dependency graphs (like the one shown in figure 1) and entailment is approximated with an alignment between the graph representing the hypothesis and a portion of the corresponding graph(s) representing the text. Each possible alignment of the graphs has an associated score, and the score of the best alignment is used as an approximation to the strength of the entailment: a better-aligned hypothesis is assumed to be more likely to be entailed. To enable incremental search, alignment scores are usually factored as a combination of local terms, corresponding to the nodes and edges of the two graphs. Unfortunately, even with factored scores the problem of finding the best alignment of two graphs is NP-complete, so exact computation is intractable. Authors have proposed a variety of approximate search techniques. Haghighi et al. (2005) divide the search into two steps: in the first step they consider node scores only, which relaxes the problem to a weighted bipartite graph matching that can be solved in polynomial time, and in the second step they add the edges scores and hillclimb the alignment via an approximate local search.

A third approach, exemplified by Moldovan et al. (2003) and Raina et al. (2005), is to translate dependency parses into neo-Davidsonian-style quasi-logical forms, and to perform weighted abductive theorem proving in the tradition of (Hobbs et al., 1988). Unless supplemented with a knowledge base, this approach is actually isomorphic to the graph matching approach. For example, the graph in figure 1 might generate the quasi-LF *rose(e1), nsubj(e1, x1), sales(x1), nn(x1, x2), Mitsubishi(x2), dobj(e1, x3), percent(x3), num(x3, x4), 46(x4)*. There is a term corresponding to each node and arc, and the resolution steps at the core of weighted abduction theorem proving consider matching an individual node of the hypothesis (e.g. *rose(e1)*) with something from the text (e.g. *fell(e1)*), just as in the graph-matching approach. The two models become distinct when there is a good supply of additional linguistic and world knowledge axioms—as in Moldovan et al. (2003) but not Raina et al. (2005). Then the theorem prover may generate intermediate forms in the proof, but, nevertheless, individual terms are resolved locally without reference to global context.

Finally, a few efforts (Akhmatova, 2005; Fowler et al., 2005; Bos and Markert, 2005) have tried to

translate sentences into formulas of first-order logic, in order to test logical entailment with a theorem prover. While in principle this approach does not suffer from the limitations we describe below, in practice it has not borne much fruit. Because few problem sentences can be accurately translated to logical form, and because logical entailment is a strict standard, recall tends to be poor.

The simple graph matching formulation of the problem belies three important issues. First, the above systems assume a form of upward monotonicity: if a good match is found with a part of the text, other material in the text is assumed not to affect the validity of the match. But many situations lack this upward monotone character. Consider variants on ID 98. Suppose the hypothesis were *Arafat targeted for assassination*. This would allow a perfect graph match or zero-cost weighted abductive proof, because the hypothesis is a subgraph of the text. However, this would be incorrect because it ignores the modal operator *could*. Information that changes the validity of a proof can also exist outside a matching clause. Consider the alternate text *Sharon denies Arafat is targeted for assassination*.[1]

The second issue is the assumption of locality. Locality is needed to allow practical search, but many entailment decisions rely on global features of the alignment, and thus do not naturally factor by nodes and edges. To take just one example, dropping a restrictive modifier preserves entailment in a positive context, but not in a negative one. For example, *Dogs barked loudly* entails *Dogs barked*, but *No dogs barked loudly* does not entail *No dogs barked*. These more global phenomena cannot be modeled with a factored alignment score.

The last issue arising in the graph matching approaches is the inherent confounding of alignment and entailment determination. The way to show that one graph element does not follow from another is to make the cost of aligning them high. However, since we are embedded in a search for the lowest cost alignment, this will just cause the system to choose an alternate alignment rather than recognizing a non-entailment. In ID 152, we would like the hypothesis to align with the first part of the text, to be able to prove that civilians are not members of law enforcement agencies and conclude that the hypothesis does not follow from the text. But a graph-matching system will to try to get non-entailment by making the matching cost between *civilians* and *members of law enforcement agencies* be very high. However, the likely result of that is that the final part of the hypothesis will align with *were civilians* at the end of the text, assuming that we allow an alignment with "loose" arc correspondence.[2] Under this candidate alignment, the lexical alignments are perfect, and the only imperfect alignment is the subject arc of *were* is mismatched in the two. A robust inference guesser will still likely conclude that there is entailment.

We propose that all three problems can be resolved in a two-stage architecture, where the alignment phase is followed by a separate phase of entailment determination. Although developed independently, the same division between alignment and classification has also been proposed by Marsi and Krahmer (2005), whose textual system is developed and evaluated on parallel translations into Dutch. Their classification phase features an output space of five semantic relations, and performs well at distinguishing entailing sentence pairs.

Finding aligned content can be done by any search procedure. Compared to previous work, we emphasize structural alignment, and seek to ignore issues like polarity and quantity, which can be left to a subsequent entailment decision. For example, the scoring function is designed to encourage antonym matches, and ignore the negation of verb predicates. The ideas clearly generalize to evaluating several alignments, but we have so far worked with just the one-best alignment. Given a good alignment, the determination of entailment reduces to a simple classification decision. The classifier is built over features designed to recognize patterns of valid and invalid inference. Weights for the features can be hand-set or chosen to minimize a relevant loss function on training data using standard techniques from machine learning. Because we already have a complete alignment, the classifier's decision can be con-

[1] This is the same problem labeled and addressed as *context* in Tatu and Moldovan (2005).

[2] Robust systems need to allow matches with imperfect arc correspondence. For instance, given *Bill went to Lyons to study French farming practices*, we would like to be able to conclude that *Bill studied French farming* despite the structural mismatch.

ditioned on arbitrary *global* features of the aligned graphs, and it can detect failures of monotonicity.

3 System

Our system has three stages: linguistic analysis, alignment, and entailment determination.

3.1 Linguistic analysis

Our goal in this stage is to compute linguistic representations of the text and hypothesis that contain as much information as possible about their semantic content. We use *typed dependency graphs*, which contain a node for each word and labeled edges representing the grammatical relations between words. Figure 1 gives the typed dependency graph for ID 971. This representation contains much of the information about words and relations between them, and is relatively easy to compute from a syntactic parse. However many semantic phenomena are not represented properly; particularly egregious is the inability to represent quantification and modality.

We parse input sentences to phrase structure trees using the Stanford parser (Klein and Manning, 2003), a statistical syntactic parser trained on the Penn TreeBank. To ensure correct parsing, we preprocess the sentences to collapse named entities into new dedicated tokens. Named entities are identified by a CRF-based NER system, similar to that described in (McCallum and Li, 2003). After parsing, contiguous collocations which appear in WordNet (Fellbaum, 1998) are identified and grouped.

We convert the phrase structure trees to typed dependency graphs using a set of deterministic hand-coded rules (de Marneffe et al., 2006). In these rules, heads of constituents are first identified using a modified version of the Collins head rules that favor semantic heads (such as lexical verbs rather than auxiliaries), and dependents of heads are typed using *tregex* patterns (Levy and Andrew, 2006), an extension of the *tgrep* pattern language. The nodes in the final graph are then annotated with their associated word, part-of-speech (given by the parser), lemma (given by a finite-state transducer described by Minnen et al. (2001)) and named-entity tag.

3.2 Alignment

The purpose of the second phase is to find a good partial alignment between the typed dependency graphs representing the hypothesis and the text. An alignment consists of a mapping from each node (word) in the hypothesis graph to a single node in the text graph, or to null.[3] Figure 1 gives the alignment for ID 971.

The space of alignments is large: there are $O((m+1)^n)$ possible alignments for a hypothesis graph with n nodes and a text graph with m nodes. We define a measure of alignment quality, and a procedure for identifying high scoring alignments. We choose a locally decomposable scoring function, such that the score of an alignment is the sum of the local node and edge alignment scores. Unfortunately, there is no polynomial time algorithm for finding the exact best alignment. Instead we use an incremental beam search, combined with a node ordering heuristic, to do approximate global search in the space of possible alignments. We have experimented with several alternative search techniques, and found that the solution quality is not very sensitive to the specific search procedure used.

Our scoring measure is designed to favor alignments which align semantically similar subgraphs, irrespective of polarity. For this reason, nodes receive high alignment scores when the words they represent are semantically similar. Synonyms and antonyms receive the highest score, and unrelated words receive the lowest. Our hand-crafted scoring metric takes into account the word, the lemma, and the part of speech, and searches for word relatedness using a range of external resources, including WordNet, precomputed latent semantic analysis matrices, and special-purpose gazettes. Alignment scores also incorporate local edge scores, which are based on the shape of the paths between nodes in the text graph which correspond to adjacent nodes in the hypothesis graph. Preserved edges receive the highest score, and longer paths receive lower scores.

3.3 Entailment determination

In the final stage of processing, we make a decision about whether or not the hypothesis is entailed by the text, conditioned on the typed dependency graphs, as well as the best alignment between them.

[3] The limitations of using one-to-one alignments are mitigated by the fact that many multiword expressions (e.g. named entities, noun compounds, multiword prepositions) have been collapsed into single nodes during linguistic analysis.

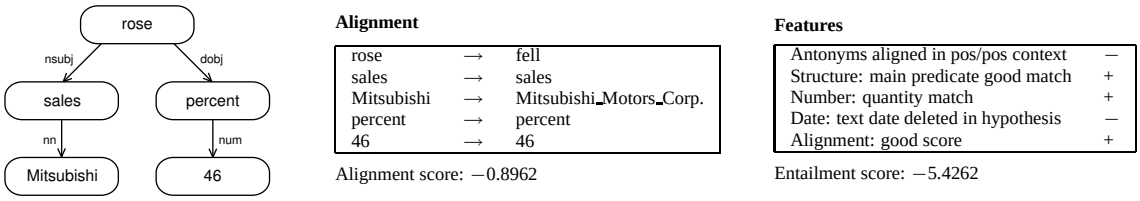


Figure 1: Problem representation for ID 971: typed dependency graph (hypothesis only), alignment, and entailment features.

Because we have a data set of examples that are labeled for entailment, we can use techniques from supervised machine learning to learn a classifier. We adopt the standard approach of defining a featural representation of the problem and then learning a linear decision boundary in the feature space. We focus here on the learning methodology; the next section covers the definition of the set of features.

Defined in this way, one can apply any statistical learning algorithm to this classification task, such as support vector machines, logistic regression, or naive Bayes. We used a logistic regression classifier with a Gaussian prior parameter for regularization. We also compare our learning results with those achieved by hand-setting the weight parameters for the classifier, effectively incorporating strong prior (human) knowledge into the choice of weights.

An advantage to the use of statistical classifiers is that they can be configured to output a probability distribution over possible answers rather than just the most likely answer. This allows us to get confidence estimates for computing a confidence weighted score (see section 5). A major concern in applying machine learning techniques to this classification problem is the relatively small size of the training set, which can lead to overfitting problems. We address this by keeping the feature dimensionality small, and using high regularization penalties in training.

4 Feature representation

In the entailment determination phase, the entailment problem is reduced to a representation as a vector of 28 features, over which the statistical classifier described above operates. These features try to capture salient patterns of entailment and non-entailment, with particular attention to contexts which reverse or block monotonicity, such as negations and quantifiers. This section describes the most important groups of features.

Polarity features. These features capture the presence (or absence) of linguistic markers of negative polarity contexts in both the text and the hypothesis, such as simple negation (*not*), downward-monotone quantifiers (*no*, *few*), restricting prepositions (*without*, *except*) and superlatives (*tallest*).

Adjunct features. These indicate the dropping or adding of syntactic adjuncts when moving from the text to the hypothesis. For the common case of restrictive adjuncts, dropping an adjunct preserves truth (*Dogs barked loudly* $\models$ *Dogs barked*), while adding an adjunct does not (*Dogs barked* $\not\models$ *Dogs barked today*). However, in negative-polarity contexts (such as *No dogs barked*), this heuristic is reversed: adjuncts can safely be added, but not dropped. For example, in ID 59, the hypothesis aligns well with the text, but the addition of *in Iraq* indicates non-entailment.

We identify the "root nodes" of the problem: the root node of the hypothesis graph and the corresponding aligned node in the text graph. Using dependency information, we identify whether adjuncts have been added or dropped. We then determine the *polarity* (negative context, positive context or restrictor of a universal quantifier) of the two root nodes to generate features accordingly.

Antonymy features. Entailment problems might involve antonymy, as in ID 971. We check whether an aligned pairs of text/hypothesis words appear to be antonymous by consulting a pre-computed list of about 40,000 antonymous and other contrasting pairs derived from WordNet. For each antonymous pair, we generate one of three boolean features, indicating whether (i) the words appear in contexts of matching polarity, (ii) only the text word appears in a negative-polarity context, or (iii) only the hypothesis word does.

Modality features. Modality features capture simple patterns of modal reasoning, as in ID 98, which illustrates the heuristic that possibility does not entail actuality. According to the occurrence (or not) of predefined modality markers, such as *must* or *maybe*, we map the text and the hypothesis to one of six modalities: *possible*, *not possible*, *actual*, *not actual*, *necessary*, and *not necessary*. The text/hypothesis modality pair is then mapped into one of the following entailment judgments: *yes*, *weak yes*, *don't know*, *weak no*, or *no*. For example:

$$(\textit{not possible} \models \textit{not actual})? \quad \Rightarrow \quad \textit{yes}$$
$$(\textit{possible} \models \textit{necessary})? \quad \Rightarrow \quad \textit{weak no}$$

Factivity features. The context in which a verb phrase is embedded may carry semantic presuppositions giving rise to (non-)entailments such as *The gangster tried to escape* $\not\models$ *The gangster escaped*. This pattern of entailment, like others, can be reversed by negative polarity markers (*The gangster managed to escape* $\models$ *The gangster escaped* while *The gangster didn't manage to escape* $\not\models$ *The gangster escaped*). To capture these phenomena, we compiled small lists of "factive" and non-factive verbs, clustered according to the kinds of entailments they create. We then determine to which class the parent of the text aligned with the hypothesis root belongs to. If the parent is not in the list, we only check whether the embedding text is an affirmative context or a negative one.

Quantifier features. These features are designed to capture entailment relations among simple sentences involving quantification, such as *Every company must report* $\models$ *A company must report* (or *The company*, or *IBM*). No attempt is made to handle multiple quantifiers or scope ambiguities. Each quantifier found in an aligned pair of text/hypothesis words is mapped into one of five quantifier categories: *no*, *some*, *many*, *most*, and *all*. The *no* category is set apart, while an ordering over the other four categories is defined. The *some* category also includes definite and indefinite determiners and small cardinal numbers. A crude attempt is made to handle negation by interchanging *no* and *all* in the presence of negation. Features are generated given the categories of both hypothesis and text.

Number, date, and time features. These are designed to recognize (mis-)matches between numbers, dates, and times, as in IDs 1806 and 231. We do some normalization (e.g. of date representations) and have a limited ability to do fuzzy matching. In ID 1806, the mismatched years are correctly identified. Unfortunately, in ID 231 the significance of *over* is not grasped and a mismatch is reported.

Alignment features. Our feature representation includes three real-valued features intended to represent the quality of the alignment: *score* is the raw score returned from the alignment phase, while *goodscore* and *badscore* try to capture whether the alignment score is "good" or "bad" by computing the sigmoid function of the distance between the alignment score and hard-coded "good" and "bad" reference values.

5 Evaluation

We present results based on the First PASCAL RTE Challenge, which used a development set containing 567 pairs and a test set containing 800 pairs. The data sets are balanced to contain equal numbers of *yes* and *no* answers. The RTE Challenge recommended two evaluation metrics: raw accuracy and confidence weighted score (CWS). The CWS is computed as follows: for each positive integer k up to the size of the test set, we compute accuracy over the k most confident predictions. The CWS is then the average, over k, of these partial accuracies. Like raw accuracy, it lies in the interval [0, 1], but it will exceed raw accuracy to the degree that predictions are well-calibrated.

Several characteristics of the RTE problems should be emphasized. Examples are derived from a broad variety of sources, including newswire; therefore systems must be domain-independent. The inferences required are, from a human perspective, fairly superficial: no long chains of reasoning are involved. However, there are "trick" questions expressly designed to foil simplistic techniques. The definition of entailment is informal and approximate: whether a competent speaker with basic knowledge of the world would typically infer the hypothesis from the text. Entailments will certainly depend on linguistic knowledge, and may also depend on world knowledge; however, the scope of required

Algorithm	RTE1 Dev Set		RTE1 Test Set	
	Acc	CWS	Acc	CWS
Random	50.0%	50.0%	50.0%	50.0%
Jijkoun et al. 05	61.0%	64.9%	55.3%	55.9%
Raina et al. 05	57.8%	66.1%	55.5%	63.8%
Haghighi et al. 05	—	—	56.8%	61.4%
Bos & Markert 05	—	—	57.7%	63.2%
Alignment only	58.7%	59.1%	54.5%	59.7%
Hand-tuned	60.3%	65.3%	**59.1%**	**65.0%**
Learning	61.2%	74.4%	**59.1%**	**63.9%**

Table 2: Performance on the RTE development and test sets. CWS stands for confidence weighted score (see text).

world knowledge is left unspecified.[4]

Despite the informality of the problem definition, human judges exhibit very good agreement on the RTE task, with agreement rate of 91–96% (Dagan et al., 2005). In principle, then, the upper bound for machine performance is quite high. In practice, however, the RTE task is exceedingly difficult for computers. Participants in the first PASCAL RTE workshop reported accuracy from 49% to 59%, and CWS from 50.0% to 69.0% (Dagan et al., 2005).

Table 2 shows results for a range of systems and testing conditions. We report accuracy and CWS on each RTE data set. The baseline for all experiments is random guessing, which always attains 50% accuracy. We show comparable results from recent systems based on lexical similarity (Jijkoun and de Rijke, 2005), graph alignment (Haghighi et al., 2005), weighted abduction (Raina et al., 2005), and a mixed system including theorem proving (Bos and Markert, 2005).

We then show results for our system under several different training regimes. The row labeled "alignment only" describes experiments in which all features except the alignment score are turned off. We predict entailment just in case the alignment score exceeds a threshold which is optimized on development data. "Hand-tuning" describes experiments in which all features are on, but no training occurs; rather, weights are set by hand, according to human intuition. Finally, "learning" describes experiments in which all features are on, and feature weights are trained on the development data. The figures reported for development data performance therefore reflect overfitting; while such results are not a fair measure of overall performance, they can help us assess the adequacy of our feature set: if our features have failed to capture relevant aspects of the problem, we should expect poor performance even when overfitting. It is therefore encouraging to see CWS above 70%. Finally, the figures reported for test data performance are the fairest basis for comparison. These are significantly better than our results for alignment only (Fisher's exact test, $p < 0.05$), indicating that we gain real value from our features. However, the gain over comparable results from other teams is not significant at the $p < 0.05$ level.

A curious observation is that the results for hand-tuned weights are as good or better than results for learned weights. A possible explanation runs as follows. Most of the features represent high-level patterns which arise only occasionally. Because the training data contains only a few hundred examples, many features are active in just a handful of instances; their learned weights are therefore quite noisy. Indeed, a feature which is expected to favor entailment may even wind up with a negative weight: the modal feature *weak yes* is an example. As shown in table 3, the learned weight for this feature was strongly negative — but this resulted from a single training example in which the feature was active but the hypothesis was not entailed. In such cases, we shouldn't expect good generalization to test data, and human intuition about the "value" of specific features may be more reliable.

Table 3 shows the values learned for selected feature weights. As expected, the features *added adjunct in* all *context*, *modal yes*, and *text is factive* were all found to be strong indicators of entailment, while *date insert*, *date modifier insert*, *widening from text to hyp* all indicate lack of entailment. Interestingly, *text has neg marker* and *text & hyp diff polarity* were also found to disfavor entailment; while this outcome is sensible, it was not anticipated or designed.

6 Conclusion

The best current approaches to the problem of textual inference work by aligning semantic graphs,

[4]Each RTE problem is also tagged as belonging to one of seven *tasks*. Previous work (Raina et al., 2005) has shown that conditioning on task can significantly improve accuracy. In this work, however, we ignore the task variable, and none of the results shown in table 2 reflect optimization by task.

Feature class	& condition	weight
Adjunct	added adjunct in *all* context	1.40
Date	date mismatch	1.30
Alignment	good score	1.10
Modal	yes	0.70
Modal	no	0.51
Factive	text is factive	0.46
...	...	...
Polarity	text & hyp same polarity	−0.45
Modal	don't know	−0.59
Quantifier	widening from text to hyp	−0.66
Polarity	text has neg marker	−0.66
Polarity	text & hyp diff polarity	−0.72
Alignment	bad score	−1.53
Date	date modifier insert	−1.57
Modal	weak yes	−1.92
Date	date insert	−2.63

Table 3: Learned weights for selected features. Positive weights favor entailment. Weights near 0 are omitted. Based on training on the PASCAL RTE development set.

using a locally-decomposable alignment score as a proxy for strength of entailment. We have argued that such models suffer from three crucial limitations: an assumption of monotonicity, an assumption of locality, and a confounding of alignment and entailment determination.

We have described a system which extends alignment-based systems while attempting to address these limitations. After finding the best alignment between text and hypothesis, we extract high-level semantic features of the entailment problem, and input these features to a statistical classifier to make an entailment decision. Using this multi-stage architecture, we report results on the PASCAL RTE data which surpass previously-reported results for alignment-based systems.

We see the present work as a first step in a promising direction. Much work remains in improving the entailment features, many of which may be seen as rough approximations to a formal monotonicity calculus. In future, we aim to combine more precise modeling of monotonicity effects with better modeling of paraphrase equivalence.

Acknowledgements

We thank Anna Rafferty, Josh Ainslie, and particularly Roger Grosse for contributions to the ideas and system reported here. This work was supported in part by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program.

References

E. Akhmatova. 2005. Textual entailment resolution via atomic propositions. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment, 2005*.

J. Bos and K. Markert. 2005. Recognising textual entailment with logical inference. In *EMNLP-05*.

I. Dagan, O. Glickman, and B. Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC 2006*.

R. de Salvo Braz, R. Girju, V. Punyakanok, D. Roth, and M. Sammons. 2005. An inference model for semantic entailment and question-answering. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI)*.

C. Fellbaum. 1998. *WordNet: an electronic lexical database*. MIT Press.

A. Fowler, B. Hauser, D. Hodges, I. Niles, A. Novischi, and J. Stephan. 2005. Applying COGEX to recognize textual entailment. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

A. Haghighi, A. Ng, and C. D. Manning. 2005. Robust textual inference via graph matching. In *EMNLP-05*.

J. R. Hobbs, M. Stickel, P. Martin, and D. D. Edwards. 1988. Interpretation as abduction. In *26th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, pages 95–103, Buffalo, New York.

V. Jijkoun and M. de Rijke. 2005. Recognizing textual entailment using lexical similarity. In *Proceedings of the PASCAL Challenge Workshop on Recognising Textual Entailment, 2005*, pages 73–76.

D. Klein and C. D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association of Computational Linguistics*.

Roger Levy and Galen Andrew. 2006. Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In *LREC 2006*.

E. Marsi and E. Krahmer. 2005. Classification of semantic relations by humans and machines. In *Proceedings of the ACL 2005 Workshop on Empirical Modeling of Semantic Equivalence and Entailment*.

A. McCallum and W. Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of CoNLL 2003*.

G. Minnen, J. Carroll, and D. Pearce. 2001. Applied morphological processing in English. In *Natural Language Engineering*, volume 7(3), pages 207–233.

D. Moldovan, C. Clark, S. Harabagiu, and S. Maiorano. 2003. COGEX: A logic prover for question answering. In *NAACL-03*.

M. Pasca and S. Harabagiu. 2001. High performance question/answering. In *SIGIR-01*, pages 366–374.

R. Raina, A .Ng, and C. D. Manning. 2005. Robust textual inference via learning and abductive reasoning. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI)*.

M. Tatu and D. Moldovan. 2005. A semantic approach to recognizing textual entailment. In *HLT/EMNLP 2005*, pages 371–378.

Acquisition of Verb Entailment from Text

Viktor Pekar
Computational Linguistics Group
University of Wolverhampton
MB109 Stafford Street
Wolverhampton WV1 1SB, UK
v.pekar@wlv.ac.uk

Abstract

The study addresses the problem of automatic acquisition of entailment relations between verbs. While this task has much in common with paraphrases acquisition which aims to discover semantic equivalence between verbs, the main challenge of entailment acquisition is to capture asymmetric, or directional, relations. Motivated by the intuition that it often underlies the local structure of coherent text, we develop a method that discovers verb entailment using evidence about discourse relations between clauses available in a parsed corpus. In comparison with earlier work, the proposed method covers a much wider range of verb entailment types and learns the mapping between verbs with highly varied argument structures.

1 Introduction

The entailment relations between verbs are a natural language counterpart of the commonsense knowledge that certain events and states give rise to other events and states. For example, there is an entailment relation between the verbs *buy* and *belong*, which reflects the commonsense notion that if someone has bought an object, this object belongs to that person.

A lexical resource encoding entailment can serve as a useful tool in many tasks where automatic inferencing over natural language text is required. In Question Answering, it has been used to establish that a certain sentence found in the corpus can serve as a suitable, albeit implicit answer to a query (Curtis et al., 2005), (Girju, 2003), (Moldovan and Rus, 2001). In Information Extraction, it can similarly help to recognize relations between named entities in cases when the entities in the text are linked by a linguistic construction that entails a known extraction pattern, but not by the pattern itself. A lexical entailment resource can contribute to information retrieval tasks via integration into a textual entailment system that aims to recognize entailment between two larger text fragments (Dagan et al., 2005).

Since entailment is known to systematically interact with the discourse organization of text (Hobbs, 1985), an entailment resource can be of interest to tasks that deal with structuring a set of individual facts into coherent text. In Natural Language Generation (Reiter and Dale, 2000) and Multi-Document Summarization (Barzilay et al., 2002) it can be used to order sentences coming from multiple, possibly unrelated sources to produce a coherent document. The knowledge is essential for compiling answers for procedural questions in a QA system, when sentences containing relevant information are spread across the corpus (Curtis et al., 2005).

The present paper is concerned with the problem of automatic acquisition of verb entailment from text. In the next section we set the background for the study by describing previous work. We then define the goal of the study and describe our method for verb entailment acquisition. After that we present results of its experimental evaluation. Finally, we draw conclusions and outline future work.

2 Previous Work

The task of verb entailment acquisition appears to have much in common with that of paraphrase acquisition (Lin and Pantel, 2001), (Pang et al., 2003), (Szpektor et al., 2004). In both tasks the goal is to discover pairs of related verbs and identify map-

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 49–56,
New York, June 2006.

pings between their argument structures. The important distinction is that while in a paraphrase the two verbs are semantically equivalent, entailment is a directional, or asymmetric, relation: one verb entails the other, but the converse does not hold. For example, the verbs *buy* and *purchase* paraphrase each other: either of them can substitute its counterpart in most contexts without altering their meaning. The verb *buy* entails *own* so that *buy* can be replaced with *own* without introducing any contradicting content into the original sentence. Replacing *own* with *buy*, however, does convey new meaning.

To account for the asymmetric character of entailment, a popular approach has been to use lexico-syntactic patterns indicative of entailment. In (Chklovski and Pantel, 2004) different types of semantic relations between verbs are discovered using surface patterns (like "*X-ed by Y-ing*" for enablement[1], which would match "*obtained by borrowing*", for example) and assessing the strength of asymmetric relations as mutual information between the two verbs. (Torisawa, 2003) collected pairs of coordinated verbs, i.e. matching patterns like "*X-ed and Y-ed*", and then estimated the probability of entailment using corpus counts. (Inui et al., 2003) used a similar approach exploiting causative expressions such as *because*, *though*, and *so*. (Girju, 2003) extracted causal relations between nouns like "*Earthquakes generate tsunami*" by first using lexico-syntactic patterns to collect relevant data and then using a decision tree classifier to learn the relations. Although these techniques have been shown to achieve high precision, their reliance on surface patterns limits their coverage in that they address only those relations that are regularly made explicit through concrete natural language expressions, and only within sentences.

The method for noun entailment acquisition by (Geffet and Dagan, 2005) is based on the idea of distributional inclusion, according to which one noun is entailed by the other if the set of occurrence contexts of the former subsumes that of the latter. However, this approach is likely to pick only a particular kind of verb entailment, that of troponymy (such as *march-walk*) and overlook pairs where there is little overlap in the occurrence patterns between the two verbs.

In tasks involving recognition of relations between entities such as Question Answering and Information Extraction, it is crucial to encode the mapping between the argument structures of two verbs. Pattern-matching often imposes restrictions on the syntactic configurations in which the verbs can appear in the corpus: the patterns employed by (Chklovski and Pantel, 2004) and (Torisawa, 2003) derive pairs of only those verbs that have identical argument structures, and often only those that involve a subject and a direct object. The method for discovery of inference rules by (Lin and Pantel, 2001) obtains pairs of verbs with highly varied argument structures, which also do not have to be identical for the two verbs. While the inference rules the method acquires seem to encompass pairs related by entailment, these pairs are not distinguished from paraphrases and the direction of relation in such pairs is not recognized.

To sum up, a major challenge in entailment acquisition is the need for more generic methods that would cover an unrestricted range of entailment types and learn the mapping between verbs with varied argument structures, eventually yielding resources suitable for robust large-scale applications.

3 Verb Entailment

Verb entailment relations have been traditionally attracting a lot of interest from lexical semantics research and their various typologies have been proposed (see, e.g., (Fellbaum, 1998)). In this study, with the view of potential practical applications, we adopt an operational definition of entailment. We define it to be a semantic relation between verbs where one verb, termed premise P, refers to event E_p and at the same time implies event E_q, typically denoted by the other verb, termed consequence Q.

The goal of verb entailment acquisition is then to find two linguistic templates each consisting of a verb and slots for its syntactic arguments. In the pair, (1) the verbs are related in accordance with our definition of entailment above, (2) there is a mapping between the slots of the two templates and (3) the direction of entailment is indicated explic-

[1] In (Chklovski and Pantel, 2004) enablement is defined to be a relation where one event often, but not necessarily always, gives rise to the other event, which coincides with our definition of entailment (see Section 3).

itly. For example, in the template pair "*buy*(*obj:X*) $\Rightarrow$ *belong*(*subj:X*)" the operator $\Rightarrow$ specifies that the premise *buy* entails the consequence *belong*, and *X* indicates a mapping between the object of *buy* and the subject of *belong*, as in *The company bought shares.* - *The shares belong to the company.*

As opposed to logical entailment, we do not require that verb entailment holds in all conceivable contexts and view it as a relation that may be more plausible in some contexts than others. For each verb pair, we therefore wish to assign a score quantifying the likelihood of its satisfying entailment in some random context.

4 Approach

The key assumption behind our approach is that the ability of a verb to imply an event typically denoted by a different verb manifests itself in the regular co-occurrence of the two verbs inside locally coherent text. This assumption is not arbitrary: as discourse investigations show (Asher and Lascarides, 2003), (Hobbs, 1985), lexical entailment plays an important role in determining the local structure of discourse. We expect this co-occurrence regularity to be equally characteristic of any pair of verbs related by entailment, regardless of is type and the syntactic behavior of verbs.

The method consists of three major steps. First, it identifies pairs of clauses that are related in the local discourse. From related clauses, it then creates templates by extracting pairs of verbs along with relevant information as to their syntactic behavior. Third, the method scores each verb pair in terms of plausibility of entailment by measuring how strongly the premise signals the appearance of the consequence inside the text segment at hand. In the following sections, we describe these steps in more detail.

4.1 Identifying discourse-related clauses

We attempt to capture local discourse relatedness between clauses by a combination of several surface cues. In doing so, we do not build a full discourse representation of text, nor do we try to identify the type of particular rhetorical relations between sentences, but rather identify pairs of clauses that are likely to be discourse-related.

Textual proximity. We start by parsing the corpus with a dependency parser (we use Connexor's FDG (Tapanainen and Järvinen, 1997)), treating every verb with its dependent constituents as a clause. For two clauses to be discourse-related, we require that they appear close to each other in the text. Adjacency of sentences has been previously used to model local coherence (Lapata, 2003). To capture related clauses within larger text fragments, we experiment with windows of text of various sizes around a clause.

Paragraph boundaries. Since locally related sentences tend to be grouped into paragraphs, we further require that the two clauses appear within the same paragraph.

Common event participant. Entity-based theories of discourse (e.g., (Grosz et al., 1995)) claim that a coherent text segment tends to focus on a specific entity. This intuition has been formalized by (Barzilay and Lapata, 2005), who developed an entity-based statistical representation of local discourse and showed its usefulness for estimating coherence between sentences. We also impose this as a criterion for two clauses to be discourse-related: their arguments need to refer to the same participant, henceforth, **anchor**. We identify the anchor as the same noun lemma appearing as an argument to the verbs in both clauses, considering only subject, object, and prepositional object arguments. The anchor must not be a pronoun, since identical pronouns may refer to different entities and making use of such correspondences is likely to introduce noise.

4.2 Creating templates

Once relevant clauses have been identified, we create pairs of syntactic templates, each consisting of a verb and the label specifying the syntactic role the anchor occupies near the verb. For example, given a pair of clauses *Mary bought a house.* and *The house belongs to Mary.*, the method will extract two pairs of templates: {*buy*(*obj:X*), *belong*(*subj:X*)} and {*buy*(*subj:X*), *belong*(*to:X*).}

Before templates are constructed, we automatically convert complex sentence parses to simpler, but semantically equivalent ones so as to increase the amount of usable data and reduce noise:

- Passive constructions are turned into active

ones: *X was bought by Y – Y bought X*;

- Phrases with coordinated nouns and verbs are decomposed: *X bought A and B – X bought A, X bought B; X bought and sold A – X bought A, X sold A.*
- Phrases with past and present participles are turned into predicate structures: *the group led by A – A leads the group; the group leading the market – the group leads the market.*

The output of this step is $V \in P \times Q$, a set of pairs of templates $\{p, q\}$, where $p \in P$ is the premise, consisting of the verb v_p and r_p – the syntactic relation between v_p and the anchor, and $q \in Q$ is the consequence, consisting of the verb v_q and r_q – its syntactic relation to the anchor.

4.3 Measuring asymmetric association

To score the pairs for asymmetric association, we use a procedure similar to the method by (Resnik, 1993) for learning selectional preferences of verbs.

Each template in a pair is tried as both a premise and a consequence. We quantify the 'preference' of the premise p for the consequence q as the contribution of q to the amount of information p contains about its consequences seen in the data. First, we calculate Kullback-Leibler Divergence (Cover. and Thomas, 1991) between two probability distributions, u – the prior distribution of all consequences in the data and w – their posterior distribution given p, thus measuring the information p contains about its consequences:

$$D_p(u||w) = \sum_n u(x) \log \frac{u(x)}{w(x)} \quad (1)$$

where $u(x) = P(q_x|p)$, $w(x) = P(q_x)$, and x ranges over all consequences in the data. Then, the score for template $\{p, q\}$ expressing the association of q with p is calculated as the proportion of q's contribution to $D_p(u||w)$:

$$Score(p, q) = P(q|p) \log \frac{P(q|p)}{P(p)} D_p(u||w)^{-1} \quad (2)$$

In each pair we compare the scores in both directions, taking the direction with the greater score to indicate the most likely premise and consequence and thus the direction of entailment.

5 Evaluation Design

5.1 Task

To evaluate the algorithm, we designed a recognition task similar to that of pseudo-word disambiguation (Schütze, 1992), (Dagan et al., 1999). The task was, given a certain premise, to select its correct consequence out of a pool with several artificially created incorrect alternatives.

The advantages of this evaluation technique are twofold. On the one hand, the task mimics many possible practical applications of the entailment resource, such as sentence ordering, where, given a sentence, it is necessary to identify among several alternatives another sentence that either entails or is entailed by the given sentence. On the other hand, in comparison with manual evaluation of the direct output of the system, it requires minimal human involvement and makes it possible to conduct large-scale experiments.

5.2 Data

The experimental material was created from the BLLIP corpus, a collection of texts from the Wall Street Journal (years 1987-89). We chose 15 transitive verbs with the greatest corpus frequency and used a pilot run of our method to extract 1000 highest-scoring template pairs involving these verbs as a premise. From them, we manually selected 129 template pairs that satisfied entailment.

For each of the 129 template pairs, four false consequences were created. This was done by randomly picking verbs with frequency comparable to that of the verb of the correct consequence. A list of parsed clauses from the BLLIP corpus was consulted to select the most typical syntactic configuration of each of the four false verbs. The resulting five template pairs, presented in a random order, constituted a test item. Figure 1 illustrates such a test item.

The entailment acquisition method was evaluated on entailment templates acquired from the British National Corpus. Even though the two corpora are quite different in style, we assume that the evaluation allows conclusions to be drawn as to the relative quality of performance of the methods under consideration.

```
1* buy(subj:X,obj:Y)⇒own(subj:X,obj:Y)
2  buy(subj:X,obj:Y)⇒approve(subj:X,obj:Y)
3  buy(subj:X,obj:Y)⇒reach(subj:X,obj:Y)
4  buy(subj:X,obj:Y)⇒decline(subj:X,obj:Y)
5  buy(subj:X,obj:Y)⇒compare(obj:X,with:Y)
```

Figure 1: An item from the test dataset. The template pair with the correct consequence is marked by an asterisk.

5.3 Recognition algorithm

During evaluation, we tested the ability of the method to select the correct consequence among the five alternatives. Our entailment acquisition method generates association scores for one-slot templates. In order to score the double-slot templates in the evaluation material, we used the following procedure.

Given a double-slot template, we divide it into two single-slot ones such that matching arguments of the two verbs along with the verbs themselves constitute a separate template. For example, "*buy* (*subj:X, obj:Y*) ⇒ *own* (*subj:X, obj:Y*)" will be decomposed into "*buy* (*subj:X*) ⇒ *own* (*subj:X*)" and "*buy* (*obj:Y*) ⇒ *own* (*obj:Y*)". The scores of these two templates are then looked up in the generated database and averaged. In each test item, the five alternatives are scored in this manner and the one with the highest score was chosen as containing the correct consequence.

The performance was measured in terms of accuracy, i.e. as the ratio of correct choices to the total number of test items. Ties, i.e. cases when the correct consequence was assigned the same score as one or more incorrect ones, contributed to the final accuracy measure proportionate to the number of tying alternatives.

This experimental design corresponds to a random baseline of 0.2, i.e. the expected accuracy when selecting a consequence template randomly out of 5 alternatives.

6 Results and Discussion

We now present the results of the evaluation of the method. In Section 6.1, we study its parameters and determine the best configuration. In Section 6.2, we compare its performance against that of human subjects as well as that of two state-of-the-art lexical resources: the verb entailment knowledge contained in WordNet2.0 and the inference rules from the DIRT database (Lin and Pantel, 2001).

6.1 Model parameters

We first examined the following parameters of the model: the window size, the use of paragraph boundaries, and the effect of the shared anchor on the quality of the model.

6.1.1 Window size and paragraph boundaries

As was mentioned in Section 4.1, a free parameter in our model is a threshold on the distance between two clauses, that we take as an indicator that the clauses are discourse-related. To find an optimal threshold, we experimented with windows of 1, 2 ... 25 clauses around a given clause, taking clauses appearing within the window as potentially related to the given one. We also looked at the effect paragraph boundaries have on the identification of related clauses. Figure 2 shows two curves depicting the accuracy of the method as a function of the window size: the first one describes performance when paragraph boundaries are taken into account (PAR) and the second one when they are ignored (NO_PAR).

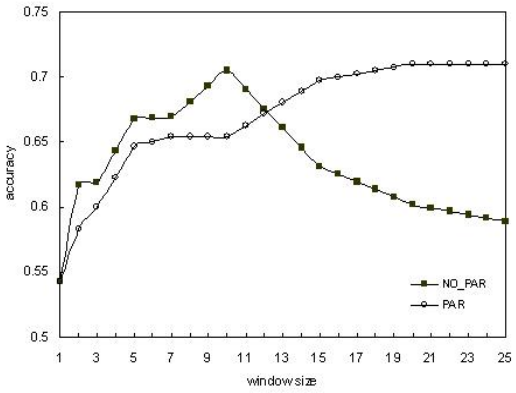


Figure 2: Accuracy of the algorithm as a function of window size, with and without paragraph boundaries used for delineating coherent text.

One can see that both curves rise fairly steeply up to window size of around 7, indicating that many entailment pairs are discovered when the two clauses appear close to each other. The rise is the steepest

between windows of 1 and 3, suggesting that entailment relations are most often explicated in clauses appearing very close to each other.

PAR reaches its maximum at the window of 15, where it levels off. Considering that 88% of paragraphs in BNC contain 15 clauses or less, we take this as an indication that a segment of text where both a premise and its consequence are likely to be found indeed roughly corresponds to a paragraph. NO_PAR's maximum is at 10, then the accuracy starts to decrease, suggesting that evidence found deeper inside other paragraphs is misleading to our model.

NO_PAR performs consistently better than PAR until it reaches its peak, i.e. when the window size is less than 10. This seems to suggest that several initial and final clauses of adjacent paragraphs are also likely to contain information useful to the model.

We tested the difference between the maxima of PAR and NO_PAR using the sign test, the non-parametric equivalent of the paired t-test. The test did not reveal any significance in the difference between their accuracies (6-, 7+, 116 ties: p = 1.000).

6.1.2 Common anchor

We further examined how the criterion of the common anchor influenced the quality of the model. We compared this model (ANCHOR) against the one that did not require that two clauses share an anchor (NO_ANCHOR), i.e. considering only co-occurrence of verbs concatenated with specific syntactic role labels. Additionally, we included into the experiment a model that looked at plain verbs co-occurring inside a context window (PLAIN). Figure 3 compares the performance of these three models (paragraph boundaries were taken into account in all of them).

Compared with ANCHOR, the other two models achieve considerably worse accuracy scores. The differences between the maximum of ANCHOR and those of the other models are significant according to the sign test (ANCHOR vs NO_ANCHOR: 44+, 8-, 77 ties: $p < 0.001$; ANCHOR vs PLAIN: 44+, 10-, 75 ties: $p < 0.001$). Their maxima are also reached sooner (at the window of 7) and thereafter their performance quickly degrades. This indicates that the common anchor criterion is very useful, especially for locating related clauses at larger distances in the text.

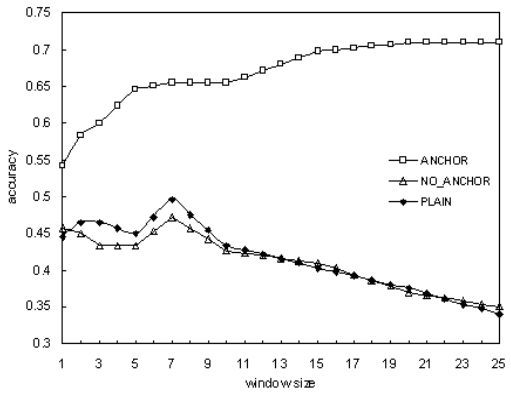


Figure 3: The effect of the common anchor on the accuracy of the method.

The accuracy scores for NO_ANCHOR and PLAIN are very similar across all the window size settings. It appears that the consistent co-occurrence of specific syntactic labels on two verbs gives no additional evidence about the verbs being related.

6.2 Human evaluation

Once the best parameter settings for the method were found, we compared its performance against human judges as well as the DIRT inference rules and the verb entailment encoded in the WordNet 2.0 database.

Human judges. To elicit human judgments on the evaluation data, we automatically converted the templates into a natural language form using a number of simple rules to arrange words in the correct grammatical order. In cases where an obligatory syntactic position near a verb was missing, we supplied the pronouns *someone* or *something* in that position. In each template pair, the premise was turned into a statement, and the consequence into a question. Figure 4 illustrates the result of converting the test item from the previous example (Figure 1) into the natural language form.

During the experiment, two judges were asked to mark those statement-question pairs in each test item, where, considering the statement, they could answer the question affirmatively. The judges' decisions coincided in 95 of 129 test items. The Kappa statistic is κ=0.725, which provides some indication about the upper bound of performance on this task.

```
X bought Y. After that:
1* Did X own Y?
2  Did X approve Y?
3  Did X reach Y?
4  Did X decline Y?
5  Did someone compare X with Y?
```

Figure 4: A test item from the test dataset. The correct consequence is marked by an asterisk.

DIRT. We also experimented with the inference rules contained in the DIRT database (Lin and Pantel, 2001). According to (Lin and Pantel, 2001), an inference rule is a relation between two verbs which are more loosely related than typical paraphrases, but nonetheless can be useful for performing inferences over natural language texts. We were interested to see how these inference rules perform on the entailment recognition task.

For each dependency tree path (a graph linking a verb with two slots for its arguments), DIRT contains a list of the most similar tree paths along with the similarity scores. To decide which is the most likely consequence in each test item, we looked up the DIRT database for the corresponding two dependency tree paths. The template pair with the greatest similarity was output as the correct answer.

WordNet. WordNet 2.0 contains manually encoded entailment relations between verb synsets, which are labeled as "cause", "troponymy", or "entailment". To identify the template pair satisfying entailment in a test item, we checked whether the two verbs in each pair are linked in WordNet in terms of one of these three labels. Because WordNet does not encode the information as to the relative plausibility of relations, all template pairs where verbs were linked in WordNet, were output as correct answers.

Figure 5 describes the accuracy scores achieved by our entailment acquisition algorithm, the two human judges, DIRT and WordNet. For comparison purposes, the random baseline is also shown.

Our algorithm outperformed WordNet by 0.38 and DIRT by 0.15. The improvement is significant vs. WordNet (73+, 27-, 29 ties: $p<0.001$) as well as vs. DIRT (37+, 20-, 72 ties: p=0.034).

We examined whether the improvement on DIRT was due to the fact that DIRT had less extensive coverage, encoding only verb pairs with similarity above a certain threshold. We re-computed the accuracy scores for the two methods, ignoring cases where DIRT did not make any decision, i.e. where the database contained none of the five verb pairs of the test item. On the resulting 102 items, our method was again at an advantage, 0.735 vs. 0.647, but the significance of the difference could not be established (21+, 12-, 69 ties: p=0.164).

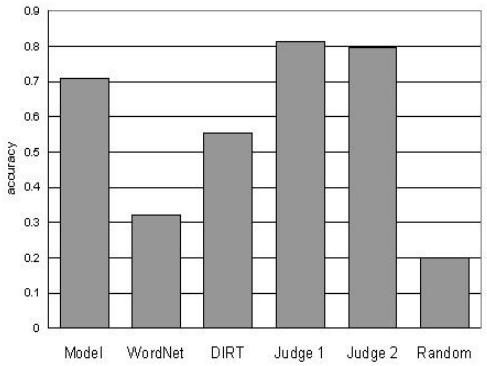


Figure 5: A comparison of performance of the proposed algorithm, WordNet, DIRT, two human judges, and a random baseline.

The difference in the performance between our algorithm and the human judges is quite large (0.103 vs. Judge 1 and 0.088 vs Judge 2), but significance to the 0.05 level could not be found (vs. Judge 1: 17-, 29+, 83 ties: p=0.105; vs. Judge 2: 15-, 27+, ties 87: p=0.09).

7 Conclusion

In this paper we proposed a novel method for automatic discovery of verb entailment relations from text, a problem that is of potential benefit for many NLP applications. The central assumption behind the method is that verb entailment relations manifest themselves in the regular co-occurrence of two verbs inside locally coherent text. Our evaluation has shown that this assumption provides a promising approach for discovery of verb entailment. The method achieves good performance, demonstrating a closer approximation to the human performance than inference rules, constructed on the basis of distributional similarity between paths in parse trees.

A promising direction along which this work

can be extended is the augmentation of the current algorithm with techniques for coreference resolution. Coreference, nominal and pronominal, is an important aspect of the linguistic realization of local discourse structure, which our model did not take into account. As the experimental evaluation suggests, many verbs related by entailment occur close to one another in the text. It is very likely that many common event participants appearing in such proximity are referred to by coreferential expressions, and therefore noticeable improvement can be expected from applying coreference resolution to the corpus prior to learning entailment patterns from it.

Acknowledgements

We are grateful to Nikiforos Karamanis and Mirella Lapata as well as three anonymous reviewers for valuable comments and suggestions. We thank Patrick Pantel and Dekang Lin for making the DIRT database available for this study.

References

N. Asher and A. Lascarides. 2003. *Logics of Conversation*. Cambridge University Press.

R. Barzilay and M. Lapata. 2005. Modeling local coherence: an entity-based approach. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 141–148.

R. Barzilay, N. Elhadad, and K. McKeown. 2002. Inferring strategies for sentence ordering in multidocument summarization. *JAIR*.

T. Chklovski and P. Pantel. 2004. VERBOCEAN: Mining the web for fine-grained semantic verb relations. In *In Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP'04)*.

T.M. Cover. and J.A. Thomas. 1991. *Elements of Information Theory*. Wiley-Interscience.

J. Curtis, G. Matthews, and D. Baxter. 2005. On the effective use of cyc in a question answering system. In *Proceedings the IJCAI'05 Workshop on Knowledge and Reasoning for Answering Questions*.

I. Dagan, L. Lee, and F. Pereira. 1999. Similarity-based models of cooccurrence probabilities. *Machine Learning*, 34(1-3):43–69.

I. Dagan, O. Glickman, and B. Magnini. 2005. The pascal recognising textual entailment challenge. In *PASCAL Challenges Workshop on Recognising Textual Entailment*.

C. Fellbaum, 1998. *WordNet: An Electronic Lexical Database*, chapter Semantic network of English verbs. MIT Press.

M. Geffet and I. Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 107–114.

R. Girju. 2003. Automatic detection of causal relations for question answering. In *Proceedings of the ACL'03 Workshop on "Multilingual Summarization and Question Answering - Machine Learning and Beyond"*.

B. Grosz, A. Joshi, and S.Weinstein. 1995. Centering : a framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

J.R. Hobbs. 1985. On the coherence and structure of discourse. Technical Report CSLI-85-37, Center for the Study of Language and Information.

T. Inui, K.Inui, and Y.Matsumoto. 2003. What kinds and amounts of causal knowledge can be acquired from text by using connective markers as clues? In *Proceedings of the 6th International Conference on Discovery Science*, pages 180–193.

M. Lapata. 2003. Probabilistic text structuring: experiments with sentence ordering. In *Proceedings of the 41rd Annual Meeting of the Association for Computational Linguistics (ACL'03)*, pages 545–552.

D. Lin and P. Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360.

D. Moldovan and V. Rus. 2001. Logic form transformation of WordNet and its applicability to question answering. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL'01)*.

B. Pang, K. Knight, and D. Marcu. 2003. Syntax-based alignment of multiple translations: extracting paraphrases and generating new sentences. In *Proceedings of HLT-NAACL'2003*.

E. Reiter and R. Dale. 2000. *Building Natural Language Generation Systems*. Cambidge University Press.

P. Resnik. 1993. *Selection and Information: A Class-Based Approach to Lexical Relationships*. Ph.D. thesis, University of Pennsylvania.

H. Schütze. 1992. Context space. In *Fall Symposium on Probabilistic Approaches to Natural Language*, pages 113–120.

I. Szpektor, H. Tanev, I. Dagan, and B. Coppola. 2004. Scaling web-based acquisition of entailment relations. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP'04)*.

P. Tapanainen and T. Järvinen. 1997. A non-projective dependency parser. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 64–71.

K. Torisawa, 2003. *Questions and Answers: Theoretical and Applied Perspectives*, chapter An unsupervised learning method for commonsensical inference rules on events. University of Utrecht.

Acquiring Inference Rules with Temporal Constraints by Using Japanese Coordinated Sentences and Noun-Verb Co-occurrences

Kentaro Torisawa
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi-shi, Ishikawa-ken, 923-1211 JAPAN
torisawa@jaist.ac.jp

Abstract

This paper shows that inference rules with temporal constraints can be acquired by using verb-verb co-occurrences in Japanese coordinated sentences and verb-noun co-occurrences. For example, our unsupervised acquisition method could obtain the inference rule "If someone enforces a law, usually someone enacts the law at the same time as or before the enforcing of the law" since the verbs "enact" and "enforce" frequently co-occurred in coordinated sentences and the verbs also frequently co-occurred with the noun "law". We also show that the accuracy of the acquisition is improved by using the occurrence frequency of a single verb, which we assume indicates how *generic* the meaning of the verb is.

1 Introduction

Our goal is to develop an unsupervised method for acquiring inference rules that describe logical implications between event occurrences. As clues to find the rules, we chose Japanese coordinated sentences, which typically report two events that occur in a certain temporal order. Of course, not every coordinated sentence necessarily expresses implications. We found, though, that reliable rules can be acquired by looking at co-occurrence frequencies between verbs in coordinated sentences and co-occurrences between verbs and nouns. For example, our method could obtain the rule "If someone enforces a law, usually someone enacts the law at the same time as or before the enforcing of the law". In our experiments, when our method produced 400 rules for 1,000 given nouns, 70% of the rules were considered proper by at least three of four human judges.

Note that the acquired inference rules pose temporal constraints on occurrences of the events described in the rules. In the "enacting-and-enforcing-law" example, the constraints were expressed by the phrase "at the same time as or before the event of". We think such temporally constrained rules should be beneficial in various types of NLP applications. The rules should allow Q&A systems to *guess* or *restrict* the time at which a certain event occurs even if they cannot directly find the time in given documents. In addition, we found that a large part of the acquired rules can be regarded as *paraphrases*, and many possible applications of paraphrases should also be target applications.

To acquire rules, our method uses a score, which is basically an approximation of the probability that particular coordinated sentences will be observed. However, it is weighted by a *bias*, which embodies our assumption that frequently observed verbs are likely to appear as the consequence of a proper inference rule. This is based on our intuition that frequently appearing verbs have a *generic* meaning and tend to describe a wide range of situations, and that natural language expressions referring to a wide range of situations are more likely to be a consequence of a proper rule than *specific* expressions describing only a narrow range of events. A similar idea relying on word co-occurrence was proposed by Geffet and Dagan (Geffet and Dagan, 2005) but our method is simpler and we expect it to be applicable to a wider range of vocabularies.

Research on the automatic acquisition of inference rules, paraphrases and entailments has received much attention. Previous attempts have used, for instance, the similarities between case frames (Lin and Pan-

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 57–64,
New York, June 2006.

tel, 2001), anchor words (Barzilay and Lee, 2003; Shinyama et al., 2002; Szepektor et al., 2004), and a web-based method (Szepektor et al., 2004; Geffet and Dagan, 2005). There is also a workshop devoted to this task (Dagan et al., 2005). The obtained accuracies have still been low, however, and we think searching for other clues, such as coordinated sentences and the bias we have just mentioned, is necessary. In addition, research has also been done on the acquisition of the temporal relations (Fujiki et al., 2003; Chklovski and Pantel, 2004) by using coordinated sentences as we did, but these works did not consider the *implications* between events.

2 Algorithm with a Simplified Score

In the following, we begin by providing an overview of our algorithm. We specify the basic steps in the algorithm and the form of the rules to be acquired. We also examine the direction of implications and temporal ordering described by the rules. After that, we describe a simplified version of the scoring function that our algorithm uses and then discuss a problem related to it. The bias mechanism, which we mentioned in the introduction, is described in the section after that.

2.1 Procedure and Generated Inference Rules

Our algorithm is given a noun as its input and produces a set of inference rules. A produced rule expresses an implication relation between two descriptions including the noun. Our basic assumptions for the acquisition can be stated as follows.

- If verbs v_1 and v_2 frequently co-occur in coordinated sentences, the verbs refer to two events that actually frequently co-occur in the real world, and a sentence including v_1 and another sentence including v_2 are good candidates to be descriptions that have an implication relation and a particular temporal order between them.
- The above tendency becomes stronger when the verbs frequently co-occur with a given noun n; i.e., if v_1 and v_2 frequently co-occur in coordinated sentences and the verbs also frequently co-occur with a noun n, a sentence including v_1 and n and another sentence including v_2 and n are good candidates to be descriptions that have an implication relation between them.

Our procedure consists of the following steps.

Step 1 Select M verbs that take a given noun n as their argument most frequently.

Step 2 For each possible pair of the selected verbs, compute the value of a scoring function that embodies our assumptions, and select the N verb pairs that have the largest score values. Note that we exclude the combination of the same verb from the pairs to be considered.

Step 3 If the score value for a verb pair is higher than a threshold θ and the verbs take n as their syntactic objects, generate an inference rule from the verb pair and the noun.

Note that we used 500 as the value of M. N was set to 4 and θ was set to various values during our experiments. Another important point is that, in Step 3, the argument positions at which the given noun can appear is restricted to syntactic objects. This was because we empirically found that the rules generated from such verb-noun pairs were relatively accurate.

Assume that a given noun is "goods" and the verb pair "sell" and "manufacture" is selected in Step 3. Then, the following rule is generated.

- If someone *sells goods*, usually someone *manufactures* the *goods* at the same time as or before the event of the *selling* of the *goods*.

Although the word "someone" occurs twice, we do not demand that it refers to the same person in both instances. It just works as a placeholder. Also note that the adverb "usually"[1] was inserted to prevent the rule from being regarded as invalid by considering situations that are logically possible but unlikely in practice.

The above rule is produced when "manufacture" and "sell" frequently co-occur in coordinated sentences such as "The company *manufactured* goods and it *sold* them". One might be puzzled because the order of the occurrences of the verbs in the coordinated sentences is reversed in the rule. The verb "sell" in the *second* (embedded) sentence/clause in the coordinated sentence appears as a verb in the precondition of the rule, while "manufacture" in the *first* (embedded) sentence/clause is the verb in the consequence.

A question then, is why we chose such an order, or such a direction of implication. There is another possibility, which might seem more straightforward. From the same coordinated sentences, we could produce the rule where the direction is reversed; i.e,., "If someone *manufactures goods*, usually someone *sells*

[1] We used "`futsuu`" as a Japanese translation.

the *goods* at the same time as or *after* the manufacturing". The difference is that the rules generated by our procedure basically *infer* a *past* event from another event, while the rules with the opposite direction have to *predict* a future event. In experiments using our development set, we observed that the rules predicting future events were often unacceptable because of the uncertainty that we usually encounter in predicting the future or achieving a *future goal*. For instance, people might do something (e.g., manufacturing) with an intention to achieve some other goal (e.g., selling) in the future. But they sometimes fail to achieve their future goal for some reason. Some manufactured goods are never sold because, for instance, they are not good enough. In our experiments, we found that the precision rates of the rules with the direction we adopted were much higher than those of the rules with the opposite direction.

2.2 Simplified Scoring Function

To be precise, a rule generated by our method has the following form, where v_{pre} and v_{con} are verbs and n is a given noun.

- If someone v_{pre} n, usually someone v_{con} the n at the same time as or before the v_{pre}-ing of the n.

We assume that all three occurrences of noun n in the rule refer to the *same entity*.

Now, we define a simplified version of our scoring function as follows.

$$BasicS(n, v_{con}, v_{pre}, arg, arg') = P_{coord}(v_{con}, v_{pre}) P_{arg'}(n|v_{pre}) P_{arg}(n|v_{con}) / P(n)^2$$

Here, $P_{coord}(v_{con}, v_{pre})$ is the probability that v_{con} and v_{pre} are observed in coordinated sentences in a way that the event described by v_{con} temporally precedes or occurs at the same time as the event described by v_{pre}. (More precisely, v_{con} and v_{pre} must be the main verbs of two conjuncts S_1 and S_2 in a Japanese coordinated sentence that is literally translated to the form "S_1 *and* S_2".) This means that in the coordinated sentences, v_{con} appears first and v_{pre} second. $P_{arg'}(n|v_{pre})$ and $P_{arg}(n|v_{con})$ are the conditional probabilities that n occupies the argument positions arg' of v_{pre} and arg of v_{con}, respectively. At the beginning, as possible argument positions, we specified five argument positions, including the syntactic object and the subject. Note that when v_{pre} and v_{con} frequently co-occur in coordinated sentences and n often becomes arguments of v_{pre} and v_{con}, the score has a large value. This means that the score embodies our assumptions for acquiring rules.

The term $P_{coord}(v_{con}, v_{pre}) P_{arg'}(n|v_{pre}) P_{arg}(n|v_{con})$ in $BasicS$ is actually an approximation of the probability $P(v_{pre}, arg', n, v_{con}, arg, n)$ that we will observe the coordinated sentences such that the two sentences/clauses in the coordinated sentence are headed by v_{pre} and v_{con} and n occupies the argument positions arg' of v_{pre} and arg of v_{con}. Another important point is that the score is divided by $P(n)^2$. This is because the probabilities such as $P_{arg}(n|v_{con})$ tend to be large for a frequently observed noun n. The division by $P(n)^2$ is done to cancel such a tendency. This division does not affect the ranking for the same noun, but, since we give a *uniform* threshold for selecting the verb pairs for distinct nouns, such *normalization* is desirable, as we confirmed in experiments using our development set.

2.3 Paraphrases and Coordinated Sentences

Thus, we have defined our algorithm and a simplified scoring function. Now let us discuss a problem that is caused by the scoring function.

As mentioned in the introduction, a large portion of the acquired rules actually consists of *paraphrases*. Here, by a paraphrase, we mean a rule consisting of two descriptions referring to an identical event. The following example is an English translation of such paraphrases obtained by our method. We think this rule is acceptable. Note that we *invented* a new English verb "clearly-write" as a translation of a Japanese verb `meiki-suru` while "write" is a translation of another Japanese verb `kaku`.

- If someone clearly-writes a phone number, usually someone writes the phone number at the same time as or before the clearly-writing of the phone number.

Note that "clearly-write" and "write" have almost the same meaning but the former is often used in texts related to legal matters. Evidently, in the above rule, "clearly-write" and "write" describe the same event, and it can be seen as a *paraphrase*. There are two types of coordinated sentence that our method can use as clues to generate the rule.

- He clearly-wrote a *phone number* and wrote the *phone number*.
- He clearly-wrote a phone number, and also wrote an address.

The first sentence is more *similar* to the inference rule than the second in the sense that the two verbs

share the same object. However, it is ridiculous because it describes the same event twice. Such a sentence is not observed frequently in corpora, and will not be used as clues to generate rules in practice.

On the other hand, we frequently observe sentences of the second type in corpora, and our method generates the paraphrases from the verb-verb co-occurrences taken from such sentences. However, there is a *mismatch* between the sentence and the acquired rule in the sense that the rule describes two events related to the same object (i.e., a phone number), while the above sentence describes two events that are related to distinct objects (i.e., a phone number and an address). Regarding this mismatch, two questions need to be addressed.

The first question is *why* our method *can* acquire the rule despite the mismatch. The answer is that our method obtains the verb-verb co-occurrence probabilities ($P_{coord}(v_{con}, v_{pre})$) and the verb-noun co-occurrence probabilities (e.g., $P_{arg}(n|v_{con})$) independently, and that the method does not check whether the two verbs share an argument.

Then the next question is why our method can acquire *accurate* paraphrases from such coordinated sentences. Though we do not have a definite answer now, our hypothesis is related to the strategy that people adopt in writing coordinated sentences. When two similar but distinct events, which *can* be described by the same verb, occur successively or at the same time, people avoid repeating the same verb to describe the two events in a single sentence. Instead they try to use distinct verbs that have similar meanings. Suppose that a person wrote his name and address. To report what she did, she may write "I clearly-wrote my name and also wrote my address" but will seldom write "I clearly-wrote my name and also clearly-wrote my address". Thus, we can expect to be able to find in coordinated sentences a large number of verb pairs consisting of two verbs with similar meanings. Note that our method tends to produce two verbs that frequently co-occur with a given noun. This also helps to produce the inference rules consisting of two semantically similar verbs.

3 Bias Mechanism

We now describe a bias used in our *full* scoring function, which significantly improves the precision. The full scoring function is defined as

$$Score(n, v_{con}, v_{pre}, arg, arg') = P_{arg}(v_{con}) BasicS(n, v_{con}, v_{pre}, arg, arg').$$

The bias is denoted as $P_{arg}(v_{con})$, which is the probability that we can observe the verb v_{con}, which is the verb in the consequence of the rule, and its argument position arg is occupied by a noun, no matter which noun actually occupies the position.

An intuitive explanation of the assumption behind this bias is that as the situation within which the description of the consequence in a rule is valid becomes wider, the rule becomes more likely to be a proper one. Consider the following rules.

- If someone *demands* a compensation payment, someone *orders* the compensation payment.
- If someone *demands* a compensation payment, someone *requests* the compensation payment.

We consider the first rule to be unacceptable while the second expresses a proper implication. The difference is the situations in which the descriptions in the consequences hold. In our view, the situations described by "*order*" are more specific than those referred to by "*request*". In other words, "*order*" holds in a smaller range of situations than "*request*". *Requesting* something can happen in any situations where there exists someone who can demand something, but *ordering* can occur only in a situations where someone in a particular social position can demand something. The basic assumption behind our bias is that rules with consequences that can be valid in a *wider* range of situations, such as "requesting a compensation payment," are more likely to be proper ones than the rules with consequences that hold in a *smaller* range of situations, such as "ordering a compensation payment".

The bias $P_{arg}(v_{con})$ was introduced to capture variations of the situations in which event descriptions are valid. We assume that frequently observed verbs form *generic* descriptions that can be valid within a wide range of events, while less frequent verbs tend to describe events that can occur in a narrower range of situations and form more specific descriptions than the frequently observed verbs. Regarding the "request-order" example, (a Japanese translation of) "request" is observed more frequently than (a Japanese translation of) "order" in corpora and this observation is consistent with our assumption. A similar idea by Geffet and Dagan (Geffet and Dagan, 2005) was proposed for capturing lexical entailment. The difference is that they relied on word co-occurrences rather than the frequency of words to measure the specificity of the semantic contents of lexical descriptions, and needed Web search to avoid data sparseness in co-occurrence

statistics. On the other hand, our method needs only simple occurrence probabilities of single verbs and we expect our method to be applicable to wider vocabulary than Geffet and Dagan's method.

The following is a more mathematical justification for the bias. According to the following discussion, $P_{arg}(v_{con})$ can be seen as a metric indicating how *easily* we can *establish* an interpretation of the rule, which is formalized as a mapping between events. In our view, if we can establish the mapping *easily*, the rule tends to be acceptable. The discussion starts from a formalization of an *interpretation* of an inference rule. Consider the rule "If exp_1 occurs, usually exp_2 occurs at the same time or before the occurrence of exp_1", where exp_1 and exp_2 are natural language expressions referring to events. In the following, we call such expressions *event descriptions* and distinguish them from an *actual event* referred to by the expressions. An actual event is called an *event instance*.

A possible interpretation of the rule is that, for any event instance e_1 that can be described by the event description exp_1 in the precondition of the rule, there always exists an event instance e_2 that can be described by the event description exp_2 in the consequence and that occurs at the same time as or before e_1 occurs. Let us write $e : exp$ if event instance e can be described by event description exp. The above interpretation can then be represented by the formula

$$\Phi : \exists f(\forall e_1(e_1 : exp_1 \rightarrow \exists e_2(e_2 = f(e_1) \wedge e_2 : exp_2)).$$

Here, the mapping f represents a temporal relation between events, and the formula $e_2 = f(e_1)$ expresses that e_2 occurs at the same time as or before e_1.

The bias $P_{arg}(v_{con})$ can be considered (an approximation of) a parameter required for computing the probability that a mapping f_{random} satisfies the requirements for f in Φ when we randomly *construct* f_{random}. The probability is denoted as $P\{e_2 : exp_2 \wedge e_2 = f_{random}(e_1) | e_1 : exp_1\}^{E_1}$ where E_1 denotes the number of events describable by exp_1. We assume that the larger this probability is, the more easily we can establish f. We can approximate $P\{e_2 : exp_2 \wedge e_2 = f_{random}(e_1) | e_1 : exp_1\}$ as $P(exp_2)$ by 1) observing that the probabilistic variables e_1 and e_2 are independent since f_{random} associates them in a completely random manner and by 2) assuming that the occurrence probability of the event instances describable by exp_2 can be approximated by the probability that exp_2 is observed in text corpora. This means that $P(exp_2)$ is one of the metrics indicating how *easily* we can establish the mapping f in Φ.

Then, the next question is what kind of expressions should be regarded as the event description exp_2. A primary candidate will be the whole sentence appearing in the consequence part of the rule to be produced. Since we specify only a verb v_{con} and its argument n in the consequence in a rule, $P(exp_2)$ can be denoted by $P_{arg}(n, v_{con})$, which is the probability that we observe the expression such that v_{con} is a head verb and n occupies an argument position arg of v_{con}. By multiplying this probability to $BasicS$ as a bias, we obtain the following scoring function.

$$Score_{cooc}(n, v_{con}, v_{pre}, arg, arg') = P_{arg}(n, v_{con}) BasicS(n, v_{con}, v_{pre}, arg, arg')$$

In our experiments, though, this score did not work well. Since $P_{arg}(n, v_{con})$ often has a small value, the problem of data sparseness seems to arise. Then, we used $P_{arg}(v_{con})$, which denotes the probability of observing sentences that contain v_{con} and its argument position arg, no matter which noun occupies arg, instead of $P_{arg}(n, v_{con})$. We multiplied the probability to $BasicS$ as a bias and obtained the following score, which is actually the scoring function we propose.

$$Score(n, v_{con}, v_{pre}, arg, arg') = P_{arg}(v_{con}) BasicS(n, v_{con}, v_{pre}, arg, arg')$$

4 Experiments

4.1 Settings

We parsed 35 years of newspaper articles (Yomiuri 87-01, Mainichi 91-99, Nikkei 90-00, 3.24GB in total) and 92.6GB of HTML documents downloaded from the WWW using an existing parser (Kanayama et al., 2000) to obtain the word (co-occurrence) frequencies. All the probabilities used in our method were estimated by maximum likelihood estimation from these frequencies. We randomly picked 600 nouns as a development set. We prepared three test sets, namely test sets A, B, and C, which consisted of 100 nouns, 250 nouns and 1,000 nouns respectively. Note that all the nouns in the test sets were randomly picked and did not have any common items with the development set. In all the experiments, four human judges checked if each produced rule was a proper one without knowing how each rule was produced.

4.2 Effects of Using Coordinated Sentences

In the first series of experiments, we compared a simplified version of our scoring function $BasicS$ with some alternative scores. This was mainly to check if coordinated sentences can improve accuracy. The alternative scores we considered

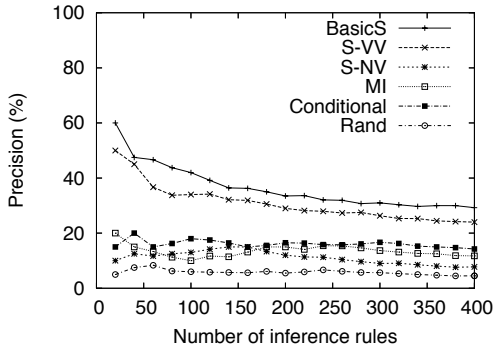


Figure 1: Comparison with the alternatives (4 judges)

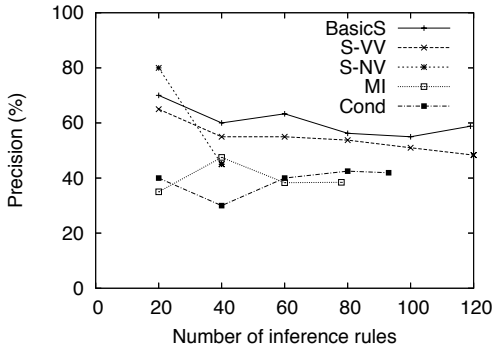


Figure 3: Comparison with the alternatives (3 judges)

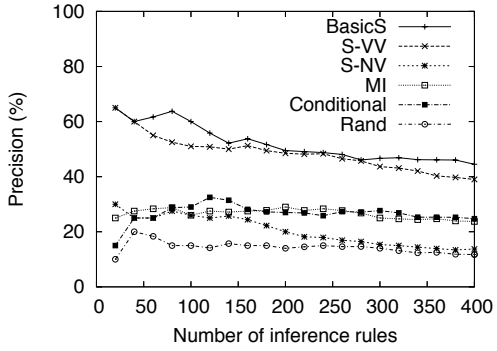


Figure 2: Comparison with the alternatives (3 judges)

are presented below. Note that we did not test our bias mechanism in this series of experiments.

$S\text{-}VV(n, v_{con}, v_{pre}, arg, arg') = P_{arg}(n, v_{con})P_{arg'}(n, v_{pre})/P(n)^2$

$S\text{-}NV(n, v_{con}, v_{pre}) = P_{coord}(v_{con}, v_{pre})$

$MI(n, v_{con}, v_{pre}) = P_{coord}(v_{con}, v_{pre})/(P(v_{con})P(v_{pre}))$

$Cond(n, v_{con}, v_{pre}, arg, arg') = P_{coord}(v_{con}, v_{pre}, arg, arg')P_{arg}(n|v_{con})P_{arg'}(n|v_{pre}) /(P_{arg'}(n, v_{pre})P(n))$

$Rand(n, v_{con}, v_{pre}, arg, arg') =$ random number

$S\text{-}VV$ was obtained by approximating the probabilities of coordinated sentences, as in the case of $BasicS$. However, we assumed the occurrences of two verbs were independent. The difference between the performance of this score and that of $BasicS$ will indicate the effectiveness of using verb-verb co-occurrences in coordinated sentences.

The second alternative, $S\text{-}NV$, simply ignores the noun-verb co-occurrences in $BasicS$. MI is a score based on mutual information and roughly corresponds to the score used in a previous attempt to acquire temporal relations between events (Chklovski and Pantel, 2004). $Cond$ is an approximation of the probability $P(n, v_{con}|n, v_{pre})$; i.e., the conditional probability that the coordinated sentences consisting of n, v_{con} and v_{pre} are observed given the precondition part consisting of v_{pre} and n. $Rand$ is a random number and generates rules by combining verbs that co-occur with the given n randomly. This was used as a baseline method of our task

The resulting precisions are shown in Figures 1 and 2. The figure captions specify "(4 judges)", as in Figure 1, when the acceptable rules included only those regarded as proper by all four judges; the captions specify "(3 judges)", as in Figure 2, when the acceptable rules include those considered proper by at least three of the four judges. We used test set A (100 nouns) and produced the top four rule candidates for each noun according to each score. As the final results, all the produced rules for all the nouns were sorted according to each score, and a precision was obtained for top N rules in the sorted list. This was the same as the precision achieved by setting the score value of N-th rule in the sorted list as threshold θ. Notice that $BasicS$ outperformed all the alternatives[2], though the difference between $S\text{-}VV$ and $BasicS$ was rather small. Another important point is that the precisions obtained with the scores that ignored noun-verb co-occurrences were quite low. These findings suggest that 1) coordinated sentences can be useful clues for obtaining temporally constrained rules and 2) noun-verb co-occurrences are also important clues.

In the above experiments, we actually allowed noun n to appear as argument types other than the syntactic objects of a verb. When we restricted the argu-

[2]Actually, the experiments concerning $Rand$ were conducted considerably after the experiments on the other scores, and only the two of the four judges for $Rand$ were included in the judges for other scores. However, we think that the superiority of our score $BasicS$ over the baseline method was confirmed since the precision of $Rand$ was drastically lower than that of $BasicS$

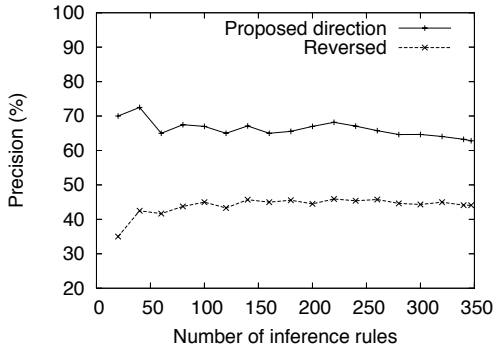


Figure 4: Two directions of implications (3 judges)

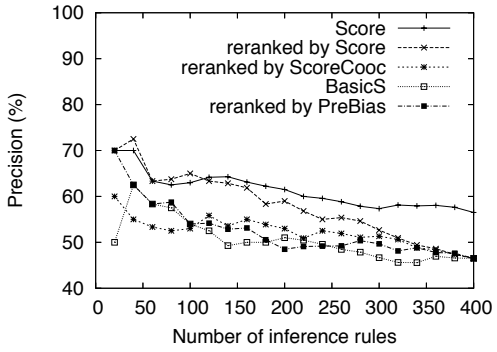


Figure 5: Effects of the bias (4 judges)

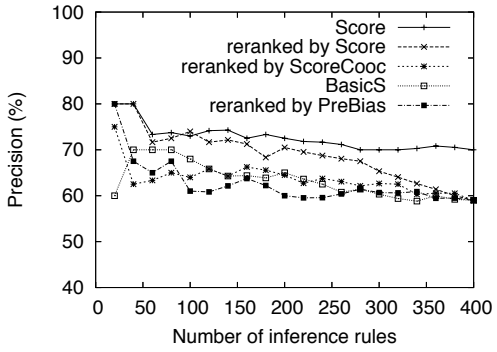


Figure 6: Effects of the bias (3 judges)

ment types to syntactic objects, as described in Section 2, the precision shown in Figure 3 was obtained. In most cases, $BasicS$ outperformed the alternatives. Although the number of produced rules was reduced because of this restriction, the precision of all produced rules was improved. Because of this, we decided to restrict the argument type to objects.

The kappa statistic for assessing the inter-rater agreement was 0.53, which indicates *moderate* agreement according to Landis and Koch, 1977. The kappa value for only the judgments on rules produced by $BasicS$ rose to 0.59. After we restricted the verb-noun co-occurrences to verb-object co-occurrences, the kappa became 0.49, while that for the rules produced by $BasicS$ was 0.54[3].

4.3 Direction of Implications

Next, we examined the directions of implications and the temporal order between events. We produced 1,000 rules for test set B (250 nouns) using the score $BasicS$, again without restricting the argument types of given nouns to syntactic objects. When we restricted the argument positions to objects, we obtained 347 rules. Then, from each generated rule, we created a new rule having an opposite direction of implications. We swapped the precondition and the consequence of the rule and reversed its temporal order. For instance, we created "If someone enacts a law, usually someone enforces the law at the same time as or *after* the enacting of the law" from "If someone enforces a law, usually someone enacts the law at the same time as or before the enforcing of the law".

Figure 4 shows the results. 'Proposed direction' refers to the precision of the rules generated by our method. The precision of the rules with the opposite direction is indicated by 'Reversed.' The precision of 'Reversed' was much lower than that of our method, and this justifies our choice of direction. The kappas values for 'BasicS' and 'Reversed' were 0.54 and 0.46 respectively. Both indicate moderate agreement.

4.4 Effects of the Bias

Last, we compared $Score$ and $BasicS$ to see the effect of our bias. This time, we used test set C (1,000 nouns). The rules were restricted to those in which the given nouns are syntactic objects of two verbs. The evaluation was done for only the top 400 rules for each score. The results are shown in Figures 5 and 6. 'Score' refers to the precision obtained with $Score$, while 'BasicS' indicates the precision with $BasicS$. For most data points in both graphs, the 'Score' precision was about 10% higher than the 'BasicS' precision. In Figure 6, the precision reached 70% when the 400 rules were produced. These results indicate the desirable effect of our bias for, at least, the top rules.

[3]These kappa values were calculated for the results except for the ones obtained by the score $Rand$, which were assessed by different judges. The kappa for $Rand$ was 0.33 (fair agreement).

rank /judges	inference rules
4/0	moshi yougi wo hininsuru naraba, yougi wo mitomeru (If someone denies suspicions, usually someone confirms the suspicions.)
6/4	moshi jikokiroku wo uwamawaru naraba, jikokiroku wo koushinsuru (If someone betters her best record, usually someone breaks her best record.)
21/3	moshi katakuriko wo mabusu naraba, katakuriko wo tsukeru (If someone coats something with potato starch, usually someone covers something with the starch)
194/4	moshi sasshi wo haifusuru naraba, sasshi wo sakuseisuru (If someone distributes a booklet, usually someone makes the booklet.)
303/4	moshi netsuzou wo kokuhakusuru naraba, netsuzou wo mitomeru (If someone confesses to a fabrication, usually someone admits the fabrication.)
398/3	moshi ifuku wo kikaeru naraba, ifuku wo nugu (If someone changes clothes, usually someone gets out of the clothes.)

Figure 7: Examples of acquired inference rules

The 400 rules generated by $Score$ included 175 distinct nouns and 272 distinct verb pairs. Examples of the inference rules acquired by $Score$ are shown in Figure 7 along with the positions in the ranking and the numbers of judges who judged the rule as being proper. (We omitted the phrase "the same time as or before" in the examples.) The kappa was 0.57 (moderate agreement).

In addition, the graphs compare $Score$ with some other alternatives. This comparison was made to check the effectiveness of our bias more carefully. The 400 rules generated by $BasicS$ were re-ranked using $Score$ and the alternative scores, and the precision for each was computed using the human judgments for the rules generated by $BasicS$. (We did not evaluate the rules directly generated by the alternatives to reduce the workload of the judges.) The first alternative was $Score_{cooc}$, which was presented in Section 3. Here, "reranked by ScoreCooc" refers to the precision obtained by re-ranking with of $Score_{cooc}$. The precision was below that obtained by the re-ranking with $Score$, (referred to as "reranked by Score)". As discussed in Section 3, this indicates the bias $P_{arg}(v_{con})$ in $Score$ works better than the bias $P_{arg}(n, v_{con})$ in $Score_{cooc}$.

The second alternative was the scoring function obtained by replacing the bias $P_{arg}(v_{con})$ in $Score$ with $P_{arg'}(v_{pre})$, which is roughly the probability that the verb in the precondition will be observed. The score is denoted as $PreBias(n, v_{con}, v_{pre}, arg, arg') = P_{arg'}(v_{pre})BasicS(n, v_{con}, v_{pre}, arg, arg')$. The precision of this score is indicated by "reranked by PreBias" and is much lower than that of "reranked by Score", indicating that only probability of the verbs in the consequences should be used as a bias. This is consistent with our assumption behind the bias.

5 Conclusion

We have presented an unsupervised method for acquiring inference rules with temporal constraints, such as "If someone enforces a law, someone enacts the law at the same time as or before the enforcing of the law". We used the probabilities of verb-verb co-occurrences in coordinated sentences and verb-noun co-occurrences. We have also proposed a bias mechanism that can improve the precision of acquired rules.

References

R. Barzilay and L. Lee. 2003. Learning to paraphrase:an unsupervised approach using multiple-sequence alignment. In *Proc. of HLT-NAACL 2003*, pages 16–23.

T. Chklovski and P. Pantel. 2004. Verbocean: Mining the web for fine-grained semantic verb relations. In *Proc. of EMNLP-04*.

I. Dagan, O. Glickman, and B. Magnini, editors. 2005. *Proceedings of the First Challenge Workshop: Recognizing Textual Entailment*. available from http://www.pascal-network.org/Challenges/RTE/.

T. Fujiki, H. Namba, and M. Okumura. 2003. Automatic acquisition of script knowledge from text collection. In *Proc. of The Research Note Sessions of EACL'03*.

M. Geffet and I. Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proc. of ACL 2005*, pages 107–114.

H. Kanayama, K. Torisawa, Y. Mitsuishi, and J. Tsujii. 2000. A hybrid Japanese parser with hand-crafted grammar and statistics. In *Proc. of COLING 2000*.

J. R. Landis and G. G. Koch. 1977. The measurement of observer agreement for categorial data. *Biometrics*, 33:159–174.

D. Lin and P. Pantel. 2001. Discovery of inference rules for question answering. *Journal of Natural Language Engineering*.

Y. Shinyama, S. Sekine, and K. Sudo. 2002. Automatic paraphrase acquisition from news articles. In *Proc. of HLT2002*.

I. Szepektor, H. Tanev, I. Dagan, and B. Coppola. 2004. Scaling web-based acquisition of entailment relations. In *Proc. of EMNLP 2004*.

Role of Local Context in Automatic Deidentification of Ungrammatical, Fragmented Text

Tawanda Sibanda
CSAIL
Massachusetts Institute of Technology
Cambridge, MA 02139
tawanda@mit.edu

Ozlem Uzuner
Department of Information Studies
College of Computing and Information
University at Albany, SUNY
Albany, NY 12222
ouzuner@albany.edu

Abstract

Deidentification of clinical records is a crucial step before these records can be distributed to non-hospital researchers. Most approaches to deidentification rely heavily on dictionaries and heuristic rules; these approaches fail to remove most personal health information (PHI) that cannot be found in dictionaries. They also can fail to remove PHI that is ambiguous between PHI and non-PHI.

Named entity recognition (NER) technologies can be used for deidentification. Some of these technologies exploit both local and global context of a word to identify its entity type. When documents are grammatically written, global context can improve NER.

In this paper, we show that we can deidentify medical discharge summaries using support vector machines that rely on a statistical representation of *local* context. We compare our approach with three different systems. Comparison with a rule-based approach shows that a statistical representation of local context contributes more to deidentification than dictionaries and hand-tailored heuristics. Comparison with two well-known systems, SNoW and IdentiFinder, shows that when the language of documents is fragmented, local context contributes more to deidentification than global context.

1 Introduction

Medical discharge summaries contain information that is useful to clinical researchers who study the interactions between, for example, different medications and diseases. However, these summaries include explicit personal health information (PHI) whose release would jeopardize privacy. In the United States, the Health Information Portability and Accountability Act (HIPAA) provides guidelines for protecting the confidentiality of health care information. HIPAA lists seventeen pieces of textual PHI of which the following appear in medical discharge summaries: first and last names of patients, their health proxies, and family members; doctors' first and last names; identification numbers; telephone, fax, and pager numbers; hospital names; geographic locations; and dates. Removing PHI from medical documents is the goal of deidentification.

This paper presents a method based on a statistical representation of *local* context for automatically removing explicit PHI from medical discharge summaries, despite the often ungrammatical, fragmented, and ad hoc language of these documents, even when some words in the documents are ambiguous between PHI and non-PHI (e.g., "Huntington" as the name of a person and as the name of a disease), and even when some of the PHI cannot be found in dictionaries (e.g., misspelled and/or foreign names). This method differs from traditional approaches to deidentification in its independence from dictionaries and hand-tailored heuristics. It applies statistical named entity recognition (NER) methods to the more challenging task of deidenti-

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 65–73,
New York, June 2006.

fication but differs from traditional NER approaches in its heavy reliance on a statistical representation of local context. Finally, this approach targets all PHI that appear in medical discharge summaries. Experiments reported in this paper show that context plays a more important role in deidentification than dictionaries, and that a statistical representation of local context contributes more to deidentification than global context.

2 Related Work

In the literature, named entities such as people, places, and organizations mentioned in news articles have been successfully identified by various approaches (Bikel et al., 1999; McCallum et al., 2000; Riloff and Jones, 1996; Collins and Singer, 1999; Hobbs et al., 1996). Most of these approaches are tailored to a particular domain, e.g., understanding disaster news; they exploit both the characteristics of the entities they focus on and the contextual clues related to these entities.

In the biomedical domain, NER has focused on identification of biological entities such as genes and proteins (Collier et al., 2000; Yu et al., 2002). Various statistical approaches, e.g., a maximum entropy model (Finkel et al., 2004), HMMs and SVMs (GuoDong et al., 2005), have been used with various feature sets including surface and syntactic features, word formation patterns, morphological patterns, part-of-speech tags, head noun triggers, and coreferences.

Deidentification refers to the removal of identifying information from records. Some approaches to deidentification have focused on particular categories of PHI, e.g., Taira et al. focused on only patient names (2002), Thomas et al. focused on proper names including doctors' names (2002). For full deidentification, i.e., removal of *all* PHI, Gupta et al. used "a complex set of rules, dictionaries, pattern-matching algorithms, and Unified Medical Language System" (2004). Sweeney's Scrub system employed competing algorithms that used patterns and lexicons to find PHI. Each of the algorithms included in her system specialized in one kind of PHI, each calculated the probability that a given word belonged to the class of PHI that it specialized in, and the algorithm with the highest precedence and the highest probability labelled the given word. This system identified 99-100% of all PHI in the test corpus of patient records and letters to physicians (1996).

We use a variety of features to train a support vector machine (SVM) that can automatically extract local context cues and can recognize PHI (even when some PHI are ambiguous between PHI and non-PHI, and even when PHI do not appear in dictionaries). We compare this approach with three others: a heuristic rule-based approach (Douglass, 2005), the SNoW (Sparse Network of Winnows) system's NER component (Roth and Yih, 2002), and IdentiFinder (Bikel et al., 1999). The heuristic rule-based system relies heavily on dictionaries. SNoW and IdentiFinder consider some representation of the local context of words; they also rely on information about global context. Local context helps them recognize stereotypical names and name structures. Global context helps these systems update the probability of observing a particular entity type based on the other entity types contained in the sentence. We hypothesize that, given the mostly fragmented and ungrammatical nature of discharge summaries, local context will be more important for deidentification than global context. We further hypothesize that local context will be a more reliable indication of PHI than dictionaries (which can be incomplete). The results presented in this paper show that SVMs trained with a statistical representation of local context outperform all baselines. In other words, a classifier that relies heavily on local context (very little on dictionaries, and not at all on global context) outperforms classifiers that rely either on global context or dictionaries (but make much less use of local context). Global context cannot contribute much to deidentification when the language of documents is fragmented; dictionaries cannot contribute to deidentification when PHI are either missing from dictionaries or are ambiguous between PHI and non-PHI. Local context remains a reliable indication of PHI under these circumstances.

The features used for our SVM-based system can be enriched in order to automatically acquire more and varied local context information. The features discussed in this paper have been chosen because of their simplicity and effectiveness on both grammatical and ungrammatical free text.

3 Corpora

Discharge summaries are the reports generated by medical personnel at the end of a patient's hospital stay and contain important information about the patient's health. Linguistic processing of these documents is challenging, mainly because these reports are full of medical jargon, acronyms, shorthand notations, misspellings, ad hoc language, and fragments of sentences. Our goal is to identify the PHI used in discharge summaries even when text is fragmented and ad hoc, even when many words in the summaries are ambiguous between PHI and non-PHI, and even when many PHI contain misspelled or foreign words.

In this study, we worked with various corpora consisting of discharge summaries. One of these corpora was obtained already deidentified[1]; i.e., (many) PHI (and some non-PHI) found in this corpus had been replaced with the generic placeholder `[REMOVED]`. An excerpt from this corpus is below:

> HISTORY OF PRESENT ILLNESS: The patient is a 77-year-old-woman with long standing hypertension who presented as a Walk-in to me at the [REMOVED] Health Center on [REMOVED]. Recently had been started q.o.d. on Clonidine since [REMOVED] to taper off of the drug. Was told to start Zestril 20 mg. q.d. again. The patient was sent to the [REMOVED] Unit for direct admission for cardioversion and anticoagulation, with the Cardiologist, Dr. [REMOVED] to follow.
>
> SOCIAL HISTORY: Lives alone, has one daughter living in [REMOVED]. Is a non-smoker, and does not drink alcohol.
>
> HOSPITAL COURSE AND TREATMENT: During admission, the patient was seen by Cardiology, Dr. [REMOVED], was started on IV Heparin, Sotalol 40 mg PO b.i.d. increased to 80 mg b.i.d., and had an echocardiogram. By [REMOVED] the patient had better rate control and blood pressure control but remained in atrial fibrillation. On [REMOVED], the patient was felt to be medically stable.
>
> ...

We hand-annotated this corpus and experimented with it in several ways: we used it to generate a corpus of discharge summaries in which the `[REMOVED]` tokens were replaced with appropriate, fake PHI obtained from dictionaries[2] (Douglass, 2005); we used it to generate a second corpus in which most of the `[REMOVED]` tokens and some of the remaining text were appropriately replaced with lexical items that were ambiguous between PHI and non-PHI[3]; we used it to generate another corpus in which all of the `[REMOVED]` tokens corresponding to names were replaced with appropriately formatted entries that could not be found in dictionaries[4]. For all of these corpora, we generated realistic substitutes for the `[REMOVED]` tokens using dictionaries (e.g., a dictionary of names from US Census Bureau) and patterns (e.g., names of people could be of the formats, "Mr. F. Lastname", "Firstname Lastname", "Lastname", "F. M. Lastname", etc.; dates could appear as "dd/mm/yy", "dd MonthName, yyyy", "ddth of MonthName, yyyy", etc.). In addition to these reidentified corpora (i.e., corpora generated from previously deidentified data), we also experimented with authentic discharge summaries[5]. The approximate distributions of PHI in the reidentified corpora and in the authentic corpus are shown in Table 1.

Class	No. in reidentified summaries	No. in authentic summaries
Non-PHI	17872	112720
Patient	1047	287
Doctor	311	730
Location	24	84
Hospital	592	651
Date	735	1933
ID	36	477
Phone	39	32

Table 1: Distribution of different PHI (in terms of number of words) in the corpora.

4 Baseline Approaches

4.1 Rule-Based Baseline: Heuristic+Dictionary

Traditional deidentification approaches rely heavily on dictionaries and hand-tailored heuristics.

[1] Authentic clinical data is very difficult to obtain for privacy reasons; therefore, the initial implementation of our system was tested on previously deidentified data that we reidentified.

[2] e.g., John Smith initiated radiation therapy ...

[3] e.g., D. Sessions initiated radiation therapy...

[4] e.g., O. Ymfgkstjj initiated radiation therapy ...

[5] We obtained authentic discharge summaries with real PHI in the final stages of this project.

We obtained one such system (Douglass, 2005) that used three kinds of dictionaries:

- PHI lookup tables for female and male first names, last names, last name prefixes, hospital names, locations, and states.
- A dictionary of "common words" that should never be classified as PHI.
- Lookup tables for context clues such as titles, e.g., Mr.; name indicators, e.g., proxy, daughter; location indicators, e.g., lives in.

Given these dictionaries, this system identifies keywords that appear in the PHI lookup tables but do not occur in the common words list, finds approximate matches for possibly misspelled words, and uses patterns and indicators to find PHI.

4.2 SNoW

SNoW is a statistical classifier that includes a NER component for recognizing entities and their relations. To create a hypothesis about the entity type of a word, SNoW first takes advantage of "words, tags, conjunctions of words and tags, bigram and trigram of words and tags", number of words in the entity, bigrams of words in the entity, and some attributes such as the prefix and suffix, as well as information about the presence of the word in a dictionary of people, organization, and location names (Roth and Yih, 2002). After this initial step, it uses the possible relations of the entity with other entities in the sentence to strengthen or weaken its hypothesis about the entity's type. The constraints imposed on the entities and their relationships constitute the global context of inference. Intuitively, information about global context and constraints imposed on the relationships of entities should improve recognition of both entities and relations. Roth and Yih (2002) present results that support this hypothesis.

SNoW can recognize entities that correspond to people, locations, and organizations. For deidentification purposes, all of these entities correspond to PHI; however, they do not constitute a comprehensive set. We evaluated SNoW only on the PHI it is built to recognize. We trained and tested its NER component using ten-fold cross-validation on each of our corpora.

4.3 IdentiFinder

IdentiFinder uses Hidden Markov Models to learn the characteristics of names of entities, including people, locations, geographic jurisdictions, organizations, dates, and contact information (Bikel et al., 1999). For each named entity class, this system learns a bigram language model which indicates the likelihood that a sequence of words belongs to that class. This model takes into consideration features of words, such as whether the word is capitalized, all upper case, or all lower case, whether it is the first word of the sentence, or whether it contains digits and punctuation. Thus, it captures the local context of the target word (i.e., the word to be classified; also referred to as TW). To find the names of all entities, the system finds the most likely sequence of entity types in a sentence given a sequence of words; thus, it captures the global context of the entities in a sentence.

We obtained this system pre-trained on a news corpus and applied it to our corpora. We mapped its entity tags to our PHI and non-PHI labels. Admittedly, testing IdentiFinder on the discharge summaries puts this system at a disadvantage compared to the other statistical approaches. However, despite this shortcoming, IdentiFinder helps us evaluate the contribution of global context to deidentification.

5 SVMs with Local Context

We hypothesize that systems that rely on dictionaries and hand-tailored heuristics face a major challenge when particular PHI can be used in many different contexts, when PHI are ambiguous, or when the PHI cannot be found in dictionaries. We further hypothesize that given the ungrammatical and ad hoc nature of our data, despite being very powerful systems, IdentiFinder and SNoW may not provide perfect deidentification. In addition to being very fragmented, discharge summaries do not present information in the form of relations between entities, and many sentences contain only one entity. Therefore, the global context utilized by IdentiFinder and SNoW cannot contribute reliably to deidentification. When run on discharge summaries, the strength of these systems comes from their ability to recognize the structure of the names of different entity types and the local contexts of these entities.

Discharge summaries contain patterns that can serve as local context. Therefore, we built an SVM-based system that, given a target word (TW), would accurately predict whether the TW was part of PHI. We used a development corpus to find features that captured as much of the immediate context of the TW as possible, paying particular attention to cues human annotators found useful for deidentification. We added to this some surface characteristics for the TW itself and obtained the following features: the TW itself, the word before, and the word after (all lemmatized); the bigram before and the bigram after TW (lemmatized); the part of speech of TW, of the word before, and of the word after; capitalization of TW; length of TW; MeSH ID of the noun phrase containing TW (MeSH is a dictionary of Medical Subject Headings and is a subset of the Unified Medical Language System (UMLS) of the National Library of Medicine); presence of TW, of the word before, and of the word after TW in the name, location, hospital, and month dictionaries; the heading of the section in which TW appears, e.g., "History of Present Illness"; and, whether TW contains "-" or "/" characters. Note that some of these features, e.g., capitalization and punctuation within TW, were also used in IdentiFinder.

We used the SVM implementation provided by LIBSVM (Chang and Lin, 2001) with a linear kernel to classify each word in the summaries as either PHI or non-PHI based on the above-listed features. We evaluated this system using ten-fold cross-validation.

6 Evaluation

Local context contributes differently to each of the four deidentification systems. Our SVM-based approach uses *only* local context. The heuristic, rule-based system relies heavily on dictionaries. IdentiFinder uses a simplified representation of local context and adds to this information about the global context as represented by transition probabilities between entities in the sentence. SNoW uses local context as well, but it also makes an effort to benefit from relations between entities. Given the difference in the strengths of these systems, we compared their performance on both the reidentified and authentic corpora (see Section 3). We hypothesized that given the nature of medical discharge summaries, IdentiFinder would not be able to find enough global context and SNoW would not be able to make use of relations (because many sentences in this corpus contain only one entity). We further hypothesized that when the data contain words ambiguous between PHI and non-PHI, or when the PHI cannot be found in dictionaries, the heuristic, rule-based approach would perform poorly. In all of these cases, SVMs trained with local context information would be sufficient for proper deidentification.

To compare the SVM approach with IdentiFinder, we evaluated both on PHI consisting of names of people (i.e., patient and doctor names), locations (i.e., geographic locations), and organizations (i.e., hospitals), as well as PHI consisting of dates, and contact information (i.e., phone numbers, pagers). We omitted PHI representing ID numbers from this experiment in order to be fair to IdentiFinder which was not trained on this category. To compare the SVM approach with SNoW, we trained both systems with only PHI consisting of names of people, locations, and organizations, i.e., the entities that SNoW was designed to recognize.

6.1 Deidentifying Reidentified and Authentic Discharge Summaries

We first deidentified:

- Previously deidentified discharge summaries into which we inserted invented but realistic surrogates for PHI without deliberately introducing ambiguous words or words not found in dictionaries, and
- Authentic discharge summaries with real PHI.

Our experiments showed that SVMs with local context outperformed all other approaches. On the reidentified corpus, SVMs gave an F-measure of 97.2% for PHI. In comparison, IdentiFinder, having been trained on the news corpus, gave an F-measure of 67.4% and was outperformed by the heuristic+dictionary approach (see Table 2).[6]

[6]Note that in deidentification, recall is much more important than precision. Low recall indicates that many PHI remain in the documents and that there is high risk to patient privacy. Low precision means that words that do not correspond to PHI have also been removed. This hurts the integrity of the data but does not present a risk to privacy.

We evaluated SNoW only on the three kinds of entities it is designed to recognize. We cross-validated it on our corpora and found that its performance in recognizing people, locations, and organizations was 96.2% in terms of F-measure (see Table 3[7]). In comparison, our SVM-based system, when retrained to only consider people, locations, and organizations so as to be directly comparable to SNoW, had an F-measure of 98%.[8]

Method	Class	P	R	F
SVM	PHI	96.8%	97.7%	**97.2%**
IFinder	PHI	60.2%	76.7%	67.4%
H+D	PHI	88.9%	67.6%	76.8%
SVM	Non-PHI	99.6%	99.5%	**99.6%**
IFinder	Non-PHI	95.8%	91.4%	93.6%
H+D	Non-PHI	95.2%	95.2%	95.2%

Table 2: Precision, Recall, and F-measure on **reidentified** discharge summaries. IFinder refers to IdentiFinder, H+D refers to heuristic+dictionary approach.

Method	Class	P	R	F
SVM	PHI	97.7%	98.2%	**98.0%**
SNoW	PHI	96.1%	96.2%	96.2%
SVM	Non-PHI	99.8%	99.8%	**99.8%**
SNoW	Non-PHI	99.6%	99.6%	99.6%

Table 3: Evaluation of SNoW and SVM on recognizing people, locations, and organizations found in **reidentified** discharge summaries.

Similarly, on the authentic discharge summaries, the SVM approach outperformed all other approaches in recognizing PHI (see Tables 4 and 5).

6.2 Deidentifying Data with Ambiguous PHI

In discharge summaries, the same words can appear both as PHI and as non-PHI. For example, in the same corpus, the word "Swan" can appear both as the name of a medical device (i.e., "Swan Catheter") and as the name of a person, etc. Ideally, we would like to deidentify data even when many words in the

[7]The best performances are marked in bold in all of the tables in this paper.

[8]For all of the corpora presented in this paper, a performance difference of 1% or more is statistically significant at $\alpha = 0.05$.

Method	Class	P	R	F
SVM	PHI	97.5%	95.0%	**96.2%**
IFinder	PHI	25.2%	45.2%	32.3%
H+D	PHI	81.9%	87.6%	84.7%
SVM	Non-PHI	99.8%	99.9%	**99.9%**
IFinder	Non-PHI	97.1%	93.3%	95.2%
H+D	Non-PHI	99.6%	99.6%	99.6%

Table 4: Evaluation on **authentic** discharge summaries.

Method	Class	P	R	F
SVM	PHI	97.4%	93.8%	**95.6%**
SNoW	PHI	93.7%	93.4%	93.6%
SVM	Non-PHI	99.9%	100%	**100%**
SNoW	Non-PHI	99.9%	99.9%	99.9%

Table 5: Evaluation of SNoW and SVM on **authentic** discharge summaries.

corpus are ambiguous between PHI and non-PHI. We hypothesize that given ambiguities in the data, context will play an important role in determining whether the particular instance of the word is PHI and that given the many fragmented sentences in our corpus, local context will be particularly useful. To test these hypotheses, we generated a corpus by reidentifying the previously deidentified corpus with words that were ambiguous between PHI and non-PHI, making sure to use each ambiguous word both as PHI and non-PHI, and also making sure to cover all acceptable formats of all PHI (see Section 3). The resulting distribution of PHI is shown in Table 6.

Class	Total # Words	# Ambiguous Words
Non-PHI	19296	3781
Patient	1047	514
Doctor	311	247
Location	24	24
Hospital	592	82
Date	736	201
ID	36	0
Phone	39	0

Table 6: Distribution of PHI when some words are ambiguous between PHI and non-PHI.

Our results showed that, on this corpus, the SVM-based system accurately recognized 91.9% of all PHI; its performance, measured in terms of F-measure was also significantly better than all other approaches both on the complete corpus containing ambiguous entries (see Table 7 and Table 8) and only on the ambiguous words in this corpus (see Table 9).

Method	Class	P	R	F
SVM	PHI	92.0%	92.1%	**92.0%**
IFinder	PHI	45.4%	71.4%	55.5%
H+D	PHI	70.1%	46.6%	56.0%
SVM	Non-PHI	98.9%	98.9%	**98.9%**
IFinder	Non-PHI	95.0%	86.5%	90.1%
H+D	Non-PHI	92.7%	92.7%	92.7%

Table 7: Evaluation **on the corpus containing ambiguous data**.

Method	Class	P	R	F
SVM	PHI	92.1%	92.8%	**92.5%**
SNoW	PHI	91.6%	77%	83.7%
SVM	Non-PHI	99.3%	99.2%	**99.3%**
SNoW	Non-PHI	97.6%	99.3%	98.4%

Table 8: Evaluation of SNoW and SVM on **ambiguous** data.

Method	Class	P	R	F
SVM	PHI	90.2%	87.5%	**88.8%**
IFinder	PHI	55.8%	64.0%	59.6%
H+D	PHI	59.8%	24.3%	34.6%
SNoW	PHI	91.6%	82.9%	87.1%
SVM	Non-PHI	90.5%	92.7%	91.6%
IFinder	Non-PHI	69.0%	61.3%	64.9%
H+D	Non-PHI	59.9%	87.4%	71.1%
SNoW	Non-PHI	90.4%	95.5%	**92.9%**

Table 9: Evaluation **only on ambiguous** people, locations, and organizations found in ambiguous data.

6.3 Deidentifying PHI Not Found in Dictionaries

Some medical documents contain foreign or misspelled names that need to be effectively removed. To evaluate the different deidentification approaches under such circumstances, we generated a corpus in which the names of people, locations, and hospitals were all random permutations of letters. The resulting words were not found in any dictionaries but followed the general format of the entity name category to which they belonged. The distribution of PHI in this third corpus is in Table 10.

Class	Total PHI	PHI Not in Dict.
Non-PHI	17872	0
Patient	1045	1045
Doctor	302	302
Location	24	24
Hospital	376	376
Date	735	0
ID	36	0
Phone	39	0

Table 10: Distribution of PHI in the corpus where all PHI associated with names are randomly generated so as not to be found in dictionaries.

On this data set, dictionaries cannot contribute to deidentification because none of the PHI appear in dictionaries. Under these conditions, proper deidentification relies completely on context. Our results showed that SVM approach outperformed all other approaches on this corpus also (Tables 11 and 12).

Method	Class	P	R	F
SVM	PHI	94.0%	96.0%	**95.0%**
IFinder	PHI	55.1%	65.5%	59.8%
H+D	PHI	76.4%	27.8%	40.8%
SVM	Non-PHI	99.4%	99.1%	**99.3%**
IFinder	Non-PHI	94.4%	91.6%	92.9%
H+D	Non-PHI	90.7%	90.7%	90.7%

Table 11: Evaluation **on the corpus containing PHI not in dictionaries**.

Of only the PHI not found in dictionaries, 95.5% was accurately identified by the SVM approach. In comparison, the heuristic+dictionary approach accurately identified those PHI that could not be found in dictionaries 11.1% of the time, IdentiFinder recognized these entities 76.7% of the time and SNoW gave an accuracy of 79% (see Table 13).

Method	Class	P	R	F
SVM	PHI	93.9%	96.0%	**95.0%**
SNoW	PHI	93.7%	79.0%	85.7%
SVM	Non-PHI	99.6%	99.4%	**99.5%**
SNoW	Non-PHI	98.0%	99.5%	98.7%

Table 12: Evaluation of SNoW and SVM on the people, locations, and organizations found in the corpus containing PHI not found in dictionaries.

Method	SVM	IFinder	SNoW	H+D
Precision	95.5%	76.7%	79.0%	11.1%

Table 13: Precision on only the PHI not found in dictionaries.

6.4 Feature Importance

As hypothesized, in all experiments, the SVM-based approach outperformed all other approaches. SVM's feature set included a total of 26 features, 12 of which were dictionary-related features (excluding MeSH). Information gain showed that the most informative features for deidentification were the TW, the bigram before TW, the bigram after TW, the word before TW, and the word after TW.

Note that the TW itself is important for classification; many of the non-PHI correspond to common words that appear in the corpus frequently and the SVM learns the fact that some words, e.g., the, admit, etc., are never PHI. In addition, the context of TW (captured in the form of unigrams and bigrams of words and part-of-speech tags surrounding TW) contributes significantly to deidentification.

There are many ways of automatically capturing context. In our data, unigrams and bigrams of words and their part-of-speech tags seem to be sufficient for a statistical representation of local context. The global context, as represented within IdentiFinder and SNoW, could not contribute much to deidentification on this corpus because of the fragmented nature of the language of these documents, because most sentences in this corpus contain only one entity, and because many sentences do not include explicit relations between entities. However, there is enough structure in this data that can be captured by local context; lack of relations between entities and the inability to capture global context do not hold us back from almost perfect deidentification.

7 Conclusion

We presented a set of experimental results that show that local context contributes more to deidentification than dictionaries and global context when working with medical discharge summaries. These documents are characterized by incomplete, fragmented sentences, and ad hoc language. They use a lot of jargon, many times omit subjects of sentences, use entity names that can be misspelled or foreign words, can include entity names that are ambiguous between PHI and non-PHI, etc. Similar documents in many domains exist; our experiments here show that even on such challenging corpora, local context can be exploited to identify entities. Even a rudimentary statistical representation of local context, as captured by unigrams and bigrams of lemmatized keywords and part-of-speech tags, gives good results and outperforms more sophisticated approaches that rely on global context. The simplicity of the representation of local context and the results obtained using this simple representation are particularly promising for many tasks that require processing ungrammatical and fragmented text where global context cannot be counted on.

8 Acknowledgements

This publication was made possible by grant number R01-EB001659 from the National Institute of Biomedical Imaging and Bioengineering; by grant number N01-LM-3-3513 on National Multi-Protocol Ensemble for Self-Scaling Systems for Health from National Library of Medicine; and, by grant number U54-LM008748 on Informatics for Integrating Biology to the Bedside from National Library of Medicine.

We are grateful to Professor Peter Szolovits and Dr. Boris Katz for their insights, and to Professor Carol Doll, Sue Felshin, Gregory Marton, and Tian He for their feedback on this paper.

References

J. J. Berman. 2002. Concept-Match Medical Data Scrubbing: How Pathology Text Can Be Used in Research. *Archives of Pathology and Laboratory Medicine*, 127(6).

D. M. Bikel, R. Schwartz, and R. M. Weischedel. 1999.

An Algorithm That Learns What's in a Name. *Machine Learning Journal Special Issue on Natural Language Learning*, 34(1/3).

C. Chang and C. Lin. 2001. *LIBSVM: a Library for Support Vector Machines*.

N. Collier, C. Nobata, and J. Tsujii. 2000. Extracting the Names of Genes and Gene Products with a Hidden Markov Model. *Proceedings of COLING*.

M. Collins and Y. Singer. 1999. Unsupervised Models for Named Entity Classification. *Proceedings of EMNLP*.

J. Finkel, S. Dingare, H. Nguyen, M. Nissim, C. Manning, and G. Sinclair. 2004. Exploiting Context for Biomedical Entity Recognition: From Syntax to the Web. *Proceedings of Joint Workshop on Natural Language Processing in Biomedicine and its Applications at COLING*.

R. Gaizauskas, G. Demetriou, P. Artymiuk, and P. Willett. 2003. Protein Structures and Information Extraction from Biological Texts: The PASTA System. *Bioinformatics*, 19(1).

Z. GuoDong, Z. Jie, S. Jian, S. Dan, T. ChewLim. 2005. Recognizing Names in Biomedical Texts: a Machine Learning Approach. *Bioinformatics*, 20(7).

D. Gupta, M. Saul, J. Gilbertson. 2004. Evaluation of a Deidentification (De-Id) Software Engine to Share Pathology Reports and Clinical Documents for Research. *American Journal of Clinical Pathology*, 121(6).

J. R. Hobbs, D. E. Appelt, J. Bear, D. Israel, M. Kameyama, M. Stickel, and M. Tyson. 1996. FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text. *In Finite State Devices for Natural Language Processing*. MIT Press, Cambridge, MA.

M. Douglass, G. D. Clifford, A. Reisner, G. B. Moody, R. G. Mark. 2005. Computer-Assisted De-Identification of Free Text in the MIMIC II Database. *Computers in Cardiology*. 32:331-334.

A. McCallum, D. Freitag, and F. Pereira. 2000. Maximum Entropy Markov Models for Information Extraction and Segmentation. *Proceedings of ICML.*

E. Riloff and R. Jones. 1996. Automatically Generating Extraction Patterns from Untagged Text. *Proceedings of AAAI-96*.

D. Roth and W. Yih. 2002. Probabilistic Reasoning for Entity and Relation Recognition. *Proceedings of COLING*.

P. Ruch, R. H. Baud, A. Rassinoux, P. Bouillon, G. Robert. 2000. Medical Document Anonymization with a Semantic Lexicon. *Proceedings of AMIA*.

M. Surdeanu, S. M. Harabagiu, J. Williams, and P. Aarseth. 2003. Using Predicate-Argument Structures for Information Extraction. *Proceedings of ACL 2003*.

L. Sweeney. 1996. Replacing personally-identifying information in medical records, the scrub system. *Journal of the American Medical Informatics Association*.

R. K. Taira, A. A. T. Bui, H. Kangarloo. 2002. Identification of patient name references within medical documents using semantic selectional restrictions. *Proceedings of AMIA*.

S. M. Thomas, B. Mamlin, G. Schadow, C. McDonald. 2002. A Successful Technique for Removing Names in Pathology Reports Using an Augmented Search and Replace Method. *Proceedings of AMIA*.

H. Yu, V. Hatzivassiloglou, C. Friedman, W. J. Wilbur. 2002. Automatic Extraction of Gene and Protein Synonyms from MEDLINE and Journal Articles. *Proceedings of AMIA*.

Exploiting Domain Structure for Named Entity Recognition

Jing Jiang and **ChengXiang Zhai**
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{jiang4,czhai}@cs.uiuc.edu

Abstract

Named Entity Recognition (NER) is a fundamental task in text mining and natural language understanding. Current approaches to NER (mostly based on supervised learning) perform well on domains similar to the training domain, but they tend to adapt poorly to slightly different domains. We present several strategies for exploiting the domain structure in the training data to learn a more robust named entity recognizer that can perform well on a new domain. First, we propose a simple yet effective way to automatically rank features based on their generalizabilities across domains. We then train a classifier with strong emphasis on the most generalizable features. This emphasis is imposed by putting a rank-based prior on a logistic regression model. We further propose a domain-aware cross validation strategy to help choose an appropriate parameter for the rank-based prior. We evaluated the proposed method with a task of recognizing named entities (genes) in biology text involving three species. The experiment results show that the new domain-aware approach outperforms a state-of-the-art baseline method in adapting to new domains, especially when there is a great difference between the new domain and the training domain.

1 Introduction

Named Entity Recognition (NER) is the task of identifying and classifying phrases that denote certain types of *named entities* (NEs), such as persons, organizations and locations in news articles, and genes, proteins and chemicals in biomedical literature. NER is a fundamental task in many natural language processing applications, such as question answering, machine translation, text mining, and information retrieval (Srihari and Li, 1999; Huang and Vogel, 2002).

Existing approaches to NER are mostly based on supervised learning. They can often achieve high accuracy provided that a large annotated training set *similar* to the test data is available (Borthwick, 1999; Zhou and Su, 2002; Florian et al., 2003; Klein et al., 2003; Finkel et al., 2005). Unfortunately, when the test data has some difference from the training data, these approaches tend to not perform well. For example, Ciaramita and Altun (2005) reported a performance degradation of a named entity recognizer trained on CoNLL 2003 Reuters corpus, where the F1 measure dropped from 0.908 when tested on a similar Reuters set to 0.643 when tested on a Wall Street Journal set. The degradation can be expected to be worse if the training data and the test data are more different.

The performance degradation indicates that existing approaches adapt poorly to new domains. We believe one reason for this poor adaptability is that these approaches have not considered the fact that, depending on the genre or domain of the text, the entities to be recognized may have different mor-

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 74–81,
New York, June 2006.

phological properties or occur in different contexts. Indeed, since most existing learning-based NER approaches explore a large feature space, without regularization, a learned NE recognizer can easily overfit the training domain.

Domain overfitting is a serious problem in NER because we often need to tag entities in completely new domains. Given any new test domain, it is generally quite expensive to obtain a large amount of labeled entity examples in that domain. As a result, in many real applications, we must train on data that do not fully resemble the test data.

This problem is especially serious in recognizing entities, in particular gene names, from biomedical literature. Gene names of one species can be quite different from those of another species syntactically due to their different naming conventions. For example, some biological species such as yeast use symbolic gene names like *tL(CAA)G3*, while some other species such as fly use descriptive gene names like *wingless*.

In this paper, we present several strategies for exploiting the domain structure in the training data to learn a more robust named entity recognizer that can perform well on a new domain. Our work is motivated by the fact that in many real applications, the training data available to us naturally falls into several domains that are similar in some aspects but different in others. For example, in biomedical literature, the training data can be naturally grouped by the biological species being discussed, while for news articles, the training data can be divided by the genre, the time, or the news agency of the articles. Our main idea is to exploit such domain structure in the training data to identify generalizable features which, presumably, are more useful for recognizing named entities in a new domain. Indeed, named entities across different domains often share certain common features, and it is these common features that are suitable for adaptation to new domains; features that only work for a particular domain would not be as useful as those working for multiple domains. In biomedical literature, for example, surrounding words such as *expression* and *encode* are strong indicators of gene mentions, regardless of the specific biological species being discussed, whereas species-specific name characteristics (e.g., prefix = "*-less*") would clearly not generalize well, and may even hurt the performance on a new domain. Similarly, in news articles, the part-of-speeches of surrounding words such as *"followed by a verb"* are more generalizable indicators of name mentions than capitalization, which might be misleading if the genre of the new domain is different; an extreme case is when every letter in the new domain is capitalized.

Based on these intuitions, we regard a feature as *generalizable* if it is useful for NER in *all* training domains, and propose a generalizability-based feature ranking method, in which we first rank the features within each training domain, and then combine the rankings to promote the features that are ranked high in all domains. We further propose a rank-based prior on logistic regression models, which puts more emphasis on the more generalizable features during the learning stage in a principled way. Finally, we present a domain-aware validation strategy for setting an appropriate parameter value for the rank-based prior. We evaluated our method on a biomedical literature data set with annotated gene names from three species, fly, mouse, and yeast, by treating one species as the new domain and the other two as the training domains. The experiment results show that the proposed method outperforms a baseline method that represents the state-of-the-art NER techniques.

The rest of the paper is organized as follows: In Section 2, we introduce a feature ranking method based on the generalizability of features across domains. In Section 3, we briefly introduce the logistic regression models for NER. We then propose a rank-based prior on logistic regression models and describe the domain-aware validation strategy in Section 4. The experiment results are presented in Section 5. Finally we discuss related work in Section 6 and conclude our work in Section 7.

2 Generalizability-Based Feature Ranking

We take a commonly used approach and treat NER as a sequential tagging problem (Borthwick, 1999; Zhou and Su, 2002; Finkel et al., 2005). Each token is assigned the tag I if it is part of an NE and the tag O otherwise. Let $\boldsymbol{x}$ denote the feature vector for a token, and let y denote the tag for $\boldsymbol{x}$. We first compute the probability $p(y|\boldsymbol{x})$ for each token, using a

learned classifier. We then apply Viterbi algorithm to assign the most likely tag sequence to a sequence of tokens, i.e., a sentence. The features we use follow the common practice in NER, including surface word features, orthographic features, POS tags, substrings, and contextual features in a local window of size 5 around the target token (Finkel et al., 2005).

As in any learning problem, feature selection may affect the NER performance significantly. Indeed, a very likely cause of the domain overfitting problem may be that the learned NE recognizer has picked up some non-generalizable features, which are not useful for a new domain. Below, we present a generalizability-based feature ranking method, which favors more generalizable features.

Formally, we assume that the training examples are divided into m subsets $T_1, T_2, \ldots, T_m$, corresponding to m different domains $D_1, D_2, \ldots, D_m$. We further assume that the test set T_{m+1} is from a new domain D_{m+1}, and this new domain shares some common features of the m training domains. Note that these are reasonable assumptions that reflect the situation in real problems.

We use *generalizability* to denote the amount of contribution a feature can make to the classification accuracy on any domain. Thus, a feature with high generalizability should be useful for classification on any domain. To identify the highly generalizable features, we must then compare their contributions to classification among different domains.

Suppose in each individual domain, the features can be ranked by their contributions to the classification accuracy. There are different feature ranking methods based on different criteria. Without loss of generality, let us use $r_T : F \rightarrow \{1, 2, \ldots, |F|\}$ to denote a ranking function that maps a feature $f \in F$ to a rank $r_T(f)$ based on a set of training examples T, where F is the set of all features, and the rank denotes the position of the feature in the final ranked list. The smaller the rank $r_T(f)$ is, the more important the feature f is in the training set T. For the m training domains, we thus have m ranking functions $r_{T_1}, r_{T_2}, \ldots, r_{T_m}$.

To identify the generalizable features across the m different domains, we propose to combine the m individual domain ranking functions in the following way. The idea is to give high ranks to features that are useful in all training domains . To achieve this goal, we first define a scoring function $s : F \rightarrow \mathbb{R}$ as follows:

$$s(f) = \min_{i=1}^{m} \frac{1}{r_{T_i}(f)}. \quad (1)$$

We then rank the features in decreasing order of their scores using the above scoring function. This is essentially to rank features according to their maximum rank $\max_i r_{T_i}(f)$ among the m domains. Let function r_{gen} return the rank of a feature in this combined, generalizability-based ranked list.

The original ranking function r_T used for individual domain feature ranking can use different criteria such as information gain or χ^2 statistic (Yang and Pedersen, 1997). In our experiments, we used a ranking function based on the model parameters of the classifier, which we will explain in Section 5.2.

Next, we need to incorporate this preference for generalizable features into the classifier. Note that because this generalizability-based feature ranking method is independent of the learning algorithm, it can be applied on top of any classifier. In this work, we choose the logistic regression classifier. One way to incorporate the feature ranking into the classifier is to select the top-k features, where k is chosen by cross validation. There are two potential problems with this *hard* feature selection approach. First, once k features are selected, they are treated equally during the learning stage, resulting in a loss of the preference among these k features. Second, this incremental feature selection approach does not consider the correlation among features. We propose an alternative way to incorporate the feature ranking into the classifier, where the preference for generalizable features is transformed into a non-uniform prior over the feature parameters in the model. This can be regarded as a *soft* feature selection approach.

3 Logistic Regression for NER

In binary logistic regression models, the probability of an observation $\boldsymbol{x}$ being classified as I is

$$p(I|\boldsymbol{x}, \boldsymbol{\beta}) = \frac{\exp(\beta_0 + \sum_{i=1}^{|F|} \beta_i x_i)}{1 + \exp(\beta_0 + \sum_{i=1}^{|F|} \beta_i x_i)} \quad (2)$$

$$= \frac{\exp(\boldsymbol{\beta} \cdot \boldsymbol{x'})}{1 + \exp(\boldsymbol{\beta} \cdot \boldsymbol{x'})}, \quad (3)$$

where β_0 is the bias weight, β_i ($1 \leq i \leq |F|$) are the weights for the features, and $\boldsymbol{x'}$ is the augmented feature vector with $x_0 = 1$. The weight vector $\boldsymbol{\beta}$ can be learned from the training examples by a maximum likelihood estimator. It is worth pointing out that logistic regression has a close relation with maximum entropy models. Indeed, when the features in a maximum entropy model are defined as conjunctions of a feature on observations only and a Kronecker delta of a class label, which is a common practice in NER, the maximum entropy model is equivalent to a logistic regression model (Finkel et al., 2005). Thus the logistic regression method we use for NER is essentially the same as the maximum entropy models used for NER in previous work.

To avoid overfitting, a zero mean Gaussian prior on the weights is usually used (Chen and Rosenfeld, 1999; Bender et al., 2003), and a maximum a posterior (MAP) estimator is used to maximize the posterior probability:

$$\hat{\boldsymbol{\beta}} = \arg\max_{\boldsymbol{\beta}} p(\boldsymbol{\beta}) \prod_{j=1}^{N} p(y_j|\boldsymbol{x_j}, \boldsymbol{\beta}), \tag{4}$$

where y_j is the true class label for $\boldsymbol{x_j}$, N is the number of training examples, and

$$p(\boldsymbol{\beta}) = \prod_{i=1}^{|F|} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp(-\frac{\beta_i^2}{2\sigma_i^2}). \tag{5}$$

In previous work, σ_i are set uniformly to the same value for all features, because there is in general no additional prior knowledge about the features.

4 Rank-Based Prior

Instead of using the same σ_i for all features, we propose a rank-based non-uniform Gaussian prior on the weights of the features so that more generalizable features get higher prior variances (i.e., low prior strength) and features on the bottom of the list get low prior variances (i.e., high prior strength). Since the prior has a zero mean, such a prior would force features on the bottom of the ranked list, which have the least generalizability, to have near-zero weights, but allow more generalizable features to be assigned higher weights during the training stage.

4.1 Transformation Function

We need to find a transformation function $h : \{1, 2, \ldots, |F|\} \rightarrow \mathbb{R}^+$ so that we can set $\sigma_i^2 = h(r_{\text{gen}}(f_i))$, where $r_{\text{gen}}(f_i)$ is the rank of feature f_i in the generalizability-based ranked feature list, as defined in Section 2. We choose the following h function because it has the desired properties as described above:

$$h(r) = \frac{a}{r^{1/b}}, \tag{6}$$

where a and b ($a, b > 0$) are parameters that control the degree of the confidence in the generalizability-based ranked feature list. Note that a corresponds to the prior variance assigned to the top-most feature in the ranked list. When b is small, the prior variance drops rapidly as the rank r increases, giving only a small number of top features high prior variances. When b is larger, there will be less discrimination among the features. When b approaches infinity, the prior becomes a uniform prior with the variance set to a for all features. If we set a small threshold τ on the variance, then we can derive that at least $m = \left(\frac{a}{\tau}\right)^b$ features have a prior variance greater than τ. Thus b is proportional to the logarithm of the number of features that are assigned a variance greater than the threshold τ when a is fixed. Figure 1 shows the h function when a is set to 20 and b is set to a set of different values.

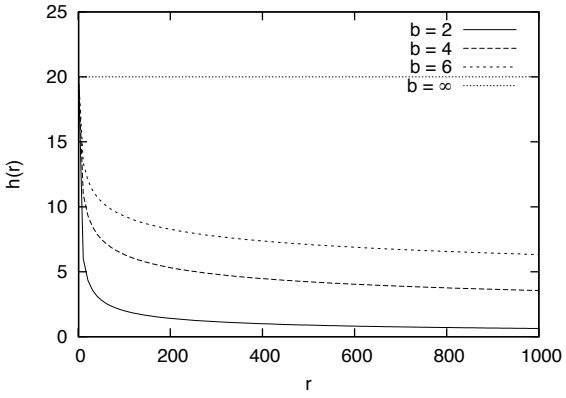


Figure 1: Transformation Function $h(r) = \frac{20}{r^{1/b}}$

4.2 Parameter Setting using Domain-Aware Validation

We need to set the appropriate values for the parameters a and b. For parameter a, we use the following

simple strategy to obtain an estimation. We first train a logistic regression model on all the training data using a Gaussian prior with a fixed variance (set to 1 in our experiments). We then find the maximum weight

$$\beta_{\max} = \max_{i=1}^{|F|} |\beta_i| \quad (7)$$

in this trained model. Finally we set $a = \beta_{\max}^2$. Our reasoning is that since a is the variance of the prior for the best feature, a is related to the "permissible range" of β for the best feature, and $\beta_{\max}$ gives us a way for adjusting a according to the empirical range of β_i's.

As we pointed out in Section 4.1, when a is fixed, parameter b controls the number of top features that are given a relatively high prior variance, and hence implicitly controls the number of top features to choose for the classifier to put the most weights on. To select an appropriate value of b, we can use a held-out validation set to tune the parameter value b. Here we present a validation strategy that exploits the domain structure in the training data to set the parameter b for a new domain. Note that in regular validation, both the training set and the validation set contain examples from all training domains. As a result, the average performance on the validation set may be dominated by domains in which the NEs are easy to classify. Since our goal is to build a classifier that performs well on new domains, we should pay more attention to hard domains that have lower classification accuracy. We should therefore examine the performance of the classifier on each training domain individually in the validation stage to gain an insight into the appropriate value of b for a new domain, which has an equal chance of being similar to any of the training domains.

Our domain-aware validation strategy first finds the optimal value of b for each training domain. For each subset T_i of the training data belonging to domain D_i, we divide it into a training set T_i^t and a validation set T_i^v. Then for each domain D_i, we train a classifier on the training sets of all domains, that is, we train on $\bigcup_{j=1}^m T_j^t$. We then test the classifier on T_i^v. We try a set of different values of b with a fixed value of a, and choose the optimal b that gives the best performance on T_i^v. Let this optimal value of b for domain D_i be b_i.

Given b_i $(1 \leq i \leq m)$, we can choose an appropriate value of b_{m+1} for an unknown test domain D_{m+1} based on the assumption that D_{m+1} is a mixture of all the training domains. b_{m+1} is then chosen to be a weighted average of b_i, $(1 \leq i \leq m)$:

$$b_{m+1} = \sum_{i=1}^{m} \lambda_i b_i, \quad (8)$$

where λ_i indicates how similar D_{m+1} is to D_i. In many cases, the test domain D_{m+1} is completely unknown. In this case, the best we can do is to set $\lambda_i = 1/m$ for all i, that is, to assume that D_{m+1} is an even mixture of all training domains.

5 Empirical Evaluation

5.1 Experimental Setup

We evaluated our domain-aware approach to NER on the problem of gene recognition in biomedical literature. The data we used is from BioCreAtIvE Task 1B (Hirschman et al., 2005). We chose this data set because it contains three subsets of MEDLINE abstracts with gene names from three species (fly, mouse, and yeast), while no other existing annotated NER data set has such explicit domain structure. The original BioCreAtIvE 1B data was not provided with every gene annotated, but for each abstract, a list of genes that were mentioned in the abstract was given. A gene synonym list was also given for each species. We used a simple string matching method with slight relaxation to tag the gene mentions in the abstracts. We took 7500 sentences from each species for our experiments, where half of the sentences contain gene mentions. We further split the 7500 sentences of each species into two sets, 5000 for training and 2500 for testing.

We conducted three sets of experiments, each combining the 5000-sentence training data of two species as training data, and the 2500-sentence test data of the third species as test data. The 2500-sentence test data of the training species was used for validation. We call these three sets of experiments *F+M⇒Y*, *F+Y⇒M*, and *M+Y⇒F*.

we use FEX[1] for feature extraction and BBR[2] for logistic regression in our experiments.

[1] http://l2r.cs.uiuc.edu/ cogcomp/asoftware.php?skey=FEX
[2] http://www.stat.rutgers.edu/ madigan/BBR/

5.2 Comparison with Baseline Method

Because the data set was generated by our automatic tagging procedure using the given gene lists, there is no previously reported performance on this data set for us to compare with. Therefore, to see whether using the domain structure in the training data can really help the adaptation to new domains, we compared our method with a state-of-the-art baseline method based on logistic regression. It uses a Gaussian prior with zero mean and uniform variance on all model parameters. It also employs 5-fold regular cross validation to pick the optimal variance for the prior. Regular feature selection is also considered in the baseline method, where the features are first ranked according to some criterion, and then cross validation is used to select the top-k features. We tested three popular regular feature ranking methods: *feature frequency* (F), *information gain* (IG), and χ^2 *statistic* (CHI). These methods were discussed in (Yang and Pedersen, 1997). However, with any of the three feature ranking criteria, cross validation showed that selecting all features gave the best average validation performance. Therefore, the best baseline method which we compare our method with uses all features. We call the baseline method *BL*.

In our method, the generalizability-based feature ranking requires a first step of feature ranking within each training domain. While we could also use F, IG or CHI to rank features in each domain, to make our method self-contained, we used the following strategy. We first train a logistic regression model on each domain using a zero-mean Gaussian prior with variance set to 1. Then, features are ranked in decreasing order of the absolute values of their weights. The rationale is that, in general, features with higher weights in the logistic regression model are more important. With this ranking within each training domain, we then use the generalizability-based feature ranking method to combine the m domain-specific rankings. The obtained ranked feature list is used to construct the rank-based prior, where the parameters a and b are set in the way as discussed in Section 4.2. We call our method *DOM*.

In Table 1, we show the precision, recall, and F1 measures of our domain-aware method (*DOM*) and the baseline method (*BL*) in all three sets of experiments. We see that the domain-aware method outperforms the baseline method in all three cases when F1 is used as the primary performance measure. In *F+Y⇒M* and *M+Y⇒F*, both precision and recall are also improved over the baseline method.

Exp	Method	P	R	F1
F+M⇒Y	BL	0.557	0.466	0.508
	DOM	0.575	0.516	**0.544**
F+Y⇒M	BL	0.571	0.335	0.422
	DOM	0.582	0.381	**0.461**
M+Y⇒F	BL	0.583	0.097	0.166
	DOM	0.591	0.139	**0.225**

Table 1: Comparison of the domain-aware method and the baseline method, where in the domain-aware method, $b = 0.5b_1 + 0.5b_2$

Note that the absolute performance shown in Table 1 is lower than the state-of-the-art performance of gene recognition (Finkel et al., 2005).[3] One reason is that we explicitly excluded the test domain from the training data, while most previous work on gene recognition was conducted on a test set drawn from the same collection as the training data. Another reason is that we used simple string matching to generate the data set, which introduced noise to the data because gene names often have irregular lexical variants.

5.3 Comparison with Regular Feature Ranking Methods

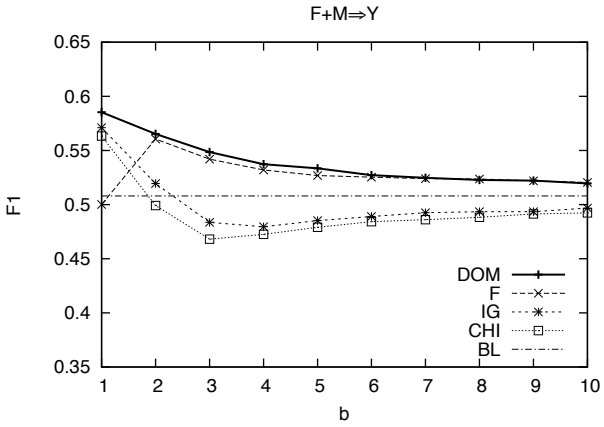


Figure 2: Comparison between regular feature ranking and generalizability-based feature ranking on *F+M⇒Y*

[3] Our baseline method performed comparably to the state-of-the-art systems on the standard BioCreAtIvE 1A data.

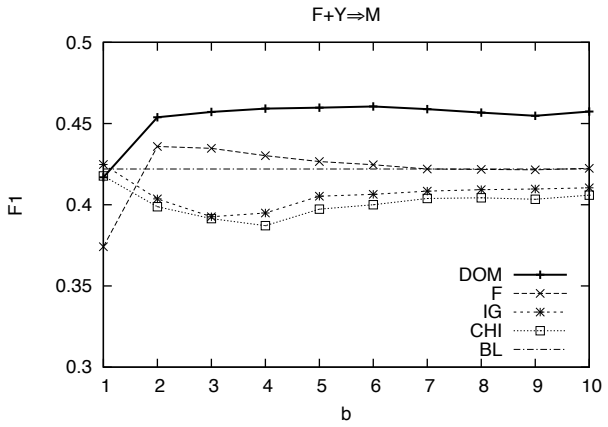


Figure 3: Comparison between regular feature ranking and generalizability-based feature ranking on $F+Y \Rightarrow M$

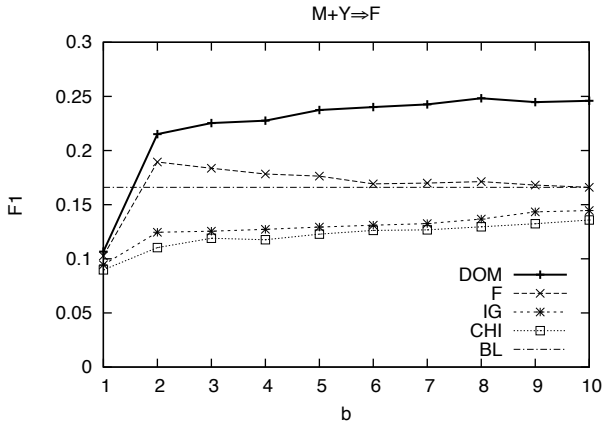


Figure 4: Comparison between regular feature ranking and generalizability-based feature ranking on $M+Y \Rightarrow F$

To further understand how our method improved the performance, we compared the generalizability-based feature ranking method with the three regular feature ranking methods, F, IG, and CHI, that were used in the baseline method. To make fair comparison, for the regular feature ranking methods, we also used the rank-based prior transformation as described in Section 4 to incorporate the preference for top-ranked features. Figure 2, Figure 3 and Figure 4 show the performance of different feature ranking methods in the three sets of experiments as the parameter b for the rank-based prior changes. As we pointed out in Section 4, b is proportional to the logarithm of the number of "effective features".

From the figures, we clearly see that the curve for the generalizability-based ranking method DOM is always above the curves of the other methods, indicating that when the same amount of top features are being emphasized by the prior, the features selected by DOM give better performance on a new domain than the features selected by the other methods. This suggests that the top-ranked features in DOM are indeed more suitable for adaptation to new domains than the top features ranked by the other methods.

The figures also show that the ranking method DOM achieved better performance than the baseline over a wide range of b values, especially in $F+Y \Rightarrow M$ and $M+Y \Rightarrow F$, whereas for methods F, IG and CHI, the performance quickly converged to the baseline performance as b increased.

It is interesting to note the comparison between F and IG (or CHI). In general, when the test data is similar to the training data, IG (or CHI) is advantageous over F (Yang and Pedersen, 1997). However, in this case when the test domain is different from the training domains, F shows advantages for adaptation. A possible explanation is that frequent features are in general less likely to be domain-specific, and therefore feature frequency can also be used as a criterion to select generalizable features and to filter out domain-specific features, although it is still not as effective as the method we proposed.

6 Related Work

The NER problem has been extensively studied in the NLP community. Most existing work has focused on supervised learning approaches, employing models such as HMMs (Zhou and Su, 2002), MEMMs (Bender et al., 2003; Finkel et al., 2005), and CRFs (McCallum and Li, 2003). Collins and Singer (1999) proposed an unsupervised method for named entity classification based on the idea of co-training. Ando and Zhang (2005) proposed a semi-supervised learning method to exploit unlabeled data for building more robust NER systems. In all these studies, the evaluation is conducted on unlabeled data similar to the labeled data.

Recently there have been some studies on adapting NER systems to new domains employing techniques such as active learning and semi-supervised learning (Shen et al., 2004; Mohit and Hwa, 2005),

or incorporating external lexical knowledge (Ciaramita and Altun, 2005). However, there has not been any study on exploiting the domain structure contained in the training examples themselves to build generalizable NER systems. We focus on the domain structure in the training data to build a classifier that relies more on features generalizable across different domains to avoid overfitting the training domains. As our method is orthogonal to most of the aforementioned work, they can be combined to further improve the performance.

7 Conclusion and Future Work

Named entity recognition is an important problem that can help many text mining and natural language processing tasks such as information extraction and question answering. Currently NER faces a poor domain adaptability problem when the test data is not from the same domain as the training data. We present several strategies to exploit the domain structure in the training data to improve the performance of the learned NER classifier on a new domain. Our results show that the domain-aware strategies we proposed improved the performance over a baseline method that represents the state-of-the-art NER techniques.

Acknowledgments

This work was in part supported by the National Science Foundation under award numbers 0425852, 0347933, and 0428472. We would like to thank Bruce Schatz, Xin He, Qiaozhu Mei, Xu Ling, and some other BeeSpace project members for useful discussions. We would like to thank Mark Sammons for his help with FEX. We would also like to thank the anonymous reviewers for their comments.

References

Rie Kubota Ando and Tong Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *Proceedings of ACL-2005*.

Oliver Bender, Franz Josef Och, and Hermann Ney. 2003. Maximum entropy models for named entity recognition. In *Proceedings of CoNLL-2003*.

Andrew Borthwick. 1999. *A Maximum Entropy Approach to Named Entity Recognition*. Ph.D. thesis, New York University.

Stanley F. Chen and Ronald Rosenfeld. 1999. A Gaussian prior for smoothing maximum entropy models. Technical Report CMU-CS-99-108, School of Computer Science, Carnegie Mellon University.

Massimiliano Ciaramita and Yasemin Altun. 2005. Named-entity recognition in novel domains with external lexical knowledge. In *Workshop on Advances in Structured Learning for Text and Speech Processing (NIPS-2005)*.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of EMNLP/VLC-1999*.

Jenny Finkel, Shipra Dingare, Christopher D. Manning, Malvina Nissim, Beatrice Alex, and Claire Grover. 2005. Exploring the boundaries: Gene and protein identification in biomedical text. *BMC Bioinformatics*, 6(Suppl 1):S5.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of CoNLL-2003*.

Lynette Hirschman, Marc Colosimo, Alexander Morgan, and Alexander Yeh. 2005. Overview of BioCreAtIvE task 1B: normailized gene lists. *BMC Bioinformatics*, 6(Suppl 1):S11.

Fei Huang and Stephan Vogel. 2002. Improved named entity translation and bilingual named entity extraction. In *ICMI-2002*.

Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher D. Manning. 2003. Named entity recognition with character-level models. In *Proceedings of CoNLL-2003*.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of CoNLL-2003*.

Behrang Mohit and Rebecca Hwa. 2005. Syntax-based semi-supervised named entity tagging. In *Proceedings of ACL-2005*.

Dan Shen, Jie Zhang, Jian Su, Guodong Zhou, and Chew-Lim Tan. 2004. Multi-criteria-based active learning for named entity recognition. In *Proceedings of ACL-2004*.

Rohini Srihari and Wei Li. 1999. Information extraction supported question answering. In *TREC-8*.

Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. In *Proceedings of ICML-1997*.

Guodong Zhou and Jian Su. 2002. Named entity recognition using an HMM-based chunk tagger. In *Proceedings of ACL-2002*.

Named Entity Transliteration and Discovery from Multilingual Comparable Corpora

Alexandre Klementiev **Dan Roth**
Department of Computer Science
University of Illinois
Urbana, IL 61801
{klementi,danr}@uiuc.edu

Abstract

Named Entity recognition (NER) is an important part of many natural language processing tasks. Most current approaches employ machine learning techniques and require supervised data. However, many languages lack such resources. This paper presents an algorithm to automatically discover Named Entities (NEs) in a resource free language, given a bilingual corpora in which it is weakly temporally aligned with a resource rich language. We observe that NEs have similar time distributions across such corpora, and that they are often transliterated, and develop an algorithm that exploits both iteratively. The algorithm makes use of a new, frequency based, metric for time distributions and a resource free discriminative approach to transliteration. We evaluate the algorithm on an English-Russian corpus, and show high level of NEs discovery in Russian.

1 Introduction

Named Entity recognition has been getting much attention in NLP research in recent years, since it is seen as a significant component of higher level NLP tasks such as information distillation and question answering, and an enabling technology for better information access. Most successful approaches to NER employ machine learning techniques, which require supervised training data. However, for many languages, these resources do not exist. Moreover, it is often difficult to find experts in these languages both for the expensive annotation effort and even for language specific clues. On the other hand, comparable multilingual data (such as multilingual news streams) are increasingly available (see section 4).

In this work, we make two independent observations about Named Entities encountered in such corpora, and use them to develop an algorithm that extracts pairs of NEs across languages. Specifically, given a bilingual corpora that is weakly temporally aligned, and a capability to annotate the text in one of the languages with NEs, our algorithm identifies the corresponding NEs in the second language text, and annotates them with the appropriate type, as in the source text.

The first observation is that NEs in one language in such corpora tend to co-occur with their counterparts in the other. E.g., Figure 1 shows a histogram of the number of occurrences of the word *Hussein* and its Russian transliteration in our bilingual news corpus spanning years 2001 through late 2005. One can see several common peaks in the two histograms, largest one being around the time of the beginning of the war in Iraq. The word *Russia*, on the other hand, has a distinctly different temporal signature. We can exploit such weak synchronicity of NEs across languages as a way to associate them. In order to score a pair of entities across languages, we compute the similarity of their time distributions.

The second observation is that NEs are often transliterated or have a common etymological origin across languages, and thus are phonetically similar. Figure 2 shows an example list of NEs and their pos-

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 82–88,
New York, June 2006.

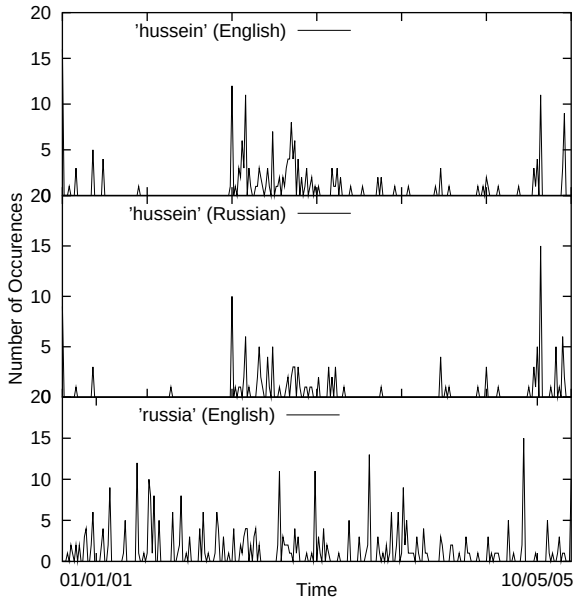


Figure 1: Temporal histograms for *Hussein* (top), its Russian transliteration (middle), and of the word *Russia* (bottom).

sible Russian transliterations.

Approaches that attempt to use these two characteristics separately to identify NEs across languages would have significant shortcomings. Transliteration based approaches require a good model, typically handcrafted or trained on a clean set of transliteration pairs. On the other hand, time sequence similarity based approaches would incorrectly match words which happen to have similar time signatures (e.g. *Taliban* and *Afghanistan* in recent news).

We introduce an algorithm we call *co-ranking* which exploits these observations simultaneously to match NEs on one side of the bilingual corpus to their counterparts on the other. We use a Discrete Fourier Transform (Arfken, 1985) based metric for computing similarity of time distributions, and we score NEs similarity with a linear transliteration model. For a given NE in one language, the transliteration model chooses a top ranked list of candidates in another language. Time sequence scoring is then used to re-rank the candidates and choose the one best temporally aligned with the NE. That is, we attempt to choose a candidate which is both a good transliteration (according to the current model) and is well aligned with the NE. Finally, pairs of NEs

English NE	Russian NE
lilic	лилич
fletcher	флетчер
bradford	брэдфорд
isabel	изабель
hoffmann	гофман
kathmandu	катманду

Figure 2: Example English NEs and their transliterated Russian counterparts.

and the best candidates are used to iteratively train the transliteration model.

A major challenge inherent in discovering transliterated NEs is the fact that a single entity may be represented by multiple transliteration strings. One reason is language morphology. For example, in Russian, depending on a case being used, the same noun may appear with various endings. Another reason is the lack of transliteration standards. Again, in Russian, several possible transliterations of an English entity may be acceptable, as long as they are phonetically similar to the source.

Thus, in order to rely on the time sequences we obtain, we need to be able to group variants of the same NE into an equivalence class, and collect their aggregate mention counts. We would then score time sequences of these equivalence classes. For instance, we would like to count the aggregate number of occurrences of {*Herzegovina*, *Hercegovina*} on the English side in order to map it accurately to the equivalence class of that NE's variants we may see on the Russian side of our corpus (e.g. {*Герцеговина, Герцеговину, Герцеговины, Герцеговиной*}).

One of the objectives for this work was to use as little of the knowledge of both languages as possible. In order to effectively rely on the quality of time sequence scoring, we used a simple, knowledge poor approach to group NE variants for Russian.

In the rest of the paper, whenever we refer to a Named Entity, we imply an NE equivalence class. Note that although we expect that better use of language specific knowledge would improve the results, it would defeat one of the goals of this work.

2 Previous Work

There has been other work to automatically discover NE with minimal supervision. Both (Cucerzan and Yarowsky, 1999) and (Collins and Singer, 1999) present algorithms to obtain NEs from untagged corpora. However, they focus on the *classification* stage of *already segmented entities*, and make use of contextual and morphological clues that require knowledge of the language beyond the level we want to assume with respect to the target language.

The use of similarity of time distributions for information extraction, in general, and NE extraction, in particular, is not new. (Hetland, 2004) surveys recent methods for scoring time sequences for similarity. (Shinyama and Sekine, 2004) used the idea to discover NEs, but in a single language, English, across two news sources.

A large amount of previous work exists on transliteration models. Most are *generative* and consider the task of *producing* an appropriate transliteration for a given word, and thus require considerable knowledge of the languages. For example, (AbdulJaleel and Larkey, 2003; Jung et al., 2000) train English-Arabic and English-Korean generative transliteration models, respectively. (Knight and Graehl, 1997) build a generative model for backward transliteration from Japanese to English.

While generative models are often robust, they tend to make independence assumptions that do not hold in data. The discriminative learning framework argued for in (Roth, 1998; Roth, 1999) as an alternative to generative models is now used widely in NLP, even in the context of word alignment (Taskar et al., 2005; Moore, 2005). We make use of it here too, to learn a discriminative transliteration model that requires little knowledge of the target language.

3 *Co-ranking*: An Algorithm for NE Discovery

In essence, the algorithm we present uses temporal alignment as a supervision signal to iteratively train a discriminative transliteration model, which can be viewed as a distance metric between and English NE and a potential transliteration. On each iteration, it selects a set of transliteration candidates for each NE according to the current model (line 6). It then uses temporal alignment (with thresholding) to select the best transliteration candidate for the next round of training (lines 8, and 9).

Once the training is complete, lines 4 through 10 are executed without thresholding for each NE in $\mathcal{S}$ to discover its counterpart in $\mathcal{T}$.

3.1 Time Sequence Generation and Matching

In order to generate time sequence for a word, we divide the corpus into a sequence of temporal bins, and count the number of occurrences of the word in each bin. We then normalize the sequence.

We use a method called the F-index (Hetland, 2004) to implement the $score$ similarity function on line 8 of the algorithm. We first run a Discrete Fourier Transform on a time sequence to extract its Fourier expansion coefficients. The score of a pair of time sequences is then computed as a Euclidian distance between their expansion coefficient vectors.

.

Input: Bilingual, comparable corpus $(\mathcal{S}, \mathcal{T})$, set of named entities $\mathcal{NE_S}$ from $\mathcal{S}$, threshold θ
Output: Transliteration model $\mathcal{M}$

1 Initialize $\mathcal{M}$;
2 $\forall \mathcal{E} \in \mathcal{NE_S}$, collect time distribution $\mathcal{Q_{ES}}$;
3 **repeat**
4 $\quad \mathcal{D} \leftarrow \emptyset$;
5 $\quad$ **for** *each* $\mathcal{E_S} \in \mathcal{NE_S}$ **do**
6 $\quad\quad$ Use $\mathcal{M}$ to collect a set of candidates $\mathcal{NE_T} \in \mathcal{T}$ with high transliteration scores;
7 $\quad\quad$ $\forall \mathcal{E} \in \mathcal{NE_T}$ collect time distribution $\mathcal{Q_{ET}}$;
8 $\quad\quad$ Select candidate $\mathcal{E_T} \in \mathcal{NE_T}$ with the best $\omega = score(\mathcal{Q_{ES}}, \mathcal{Q_{ET}})$;
9 $\quad\quad$ if ω exceeds θ, add tuple $(\mathcal{E_S}, \mathcal{E_T})$ to $\mathcal{D}$;
10 $\quad$ **end**
11 $\quad$ Use $\mathcal{D}$ to train $\mathcal{M}$;
12 **until** *D stops changing between iterations* ;

Algorithm 1: Co-ranking: an algorithm for iterative cross lingual NE discovery.

3.1.1 Equivalence Classes

As we mentioned earlier, an NE in one language may map to multiple morphological variants and transliterations in another. Identification of the entity's equivalence class of transliterations is important for obtaining its accurate time sequence.

In order to keep to our objective of requiring as little language knowledge as possible, we took a rather simplistic approach to take into account morpholog-

ical ambiguities of NEs in Russian. Two words were considered variants of the same NE if they share a prefix of size five or longer. At this point, our algorithm takes a simplistic approach also for the English side of the corpus – each unique word had its own equivalence class although, in principle, we can incorporate works such as (Li et al., 2004) into the algorithm. A cumulative distribution was then collected for such equivalence classes.

3.2 Transliteration Model

Unlike most of the previous work to transliteration, that consider *generative* transliteration models, we take a *discriminative* approach. We train a linear model to decide whether a word $\mathcal{E}_\mathcal{T} \in \mathcal{T}$ is a transliteration of an NE $\mathcal{E}_\mathcal{S} \in \mathcal{S}$. The words in the pair are partitioned into a set of substrings $s_\mathcal{S}$ and $s_\mathcal{T}$ up to a particular length (including the empty string _). Couplings of the substrings $(s_\mathcal{S}, s_\mathcal{T})$ from both sets produce features we use for training. Note that couplings with the empty string represent insertions/omissions.

Consider the following example: $(\mathcal{E}_\mathcal{S}, \mathcal{E}_\mathcal{T})$ = (powell, pauel). We build a feature vector from this example in the following manner:

- First, we split both words into all possible substrings of up to size two:

  $\mathcal{E}_\mathcal{S} \rightarrow \{\_, p, o, w, e, l, l, po, ow, we, el, ll\}$

  $\mathcal{E}_\mathcal{T} \rightarrow \{\_, p, a, u, e, l, pa, au, ue, el\}$

- We build a feature vector by coupling substrings from the two sets:

  $((p, \_), (p, a), ...(w, au), ...(el, el), ...(ll, el))$

We use the observation that transliteration tends to preserve phonetic sequence to limit the number of couplings. For example, we can disallow the coupling of substrings whose starting positions are too far apart: thus, we might not consider a pairing (po, ue) in the above example. In our experiments, we paired substrings if their positions in their respective words differed by -1, 0, or 1.

We use the perceptron (Rosenblatt, 1958) algorithm to train the model. The model activation provides the score we use to select best transliterations on line 6. Our version of perceptron takes examples with a variable number of features; each example is a set of all features seen so far that are active in the input. As the iterative algorithm observes more data, it discovers and makes use of more features. This model is called the infinite attribute model (Blum, 1992) and it follows the perceptron version in SNoW (Roth, 1998).

Positive examples used for iterative training are pairs of NEs and their best temporally aligned (thresholded) transliteration candidates. Negative examples are English non-NEs paired with random Russian words.

4 Experimental Study

We ran experiments using a bilingual comparable English-Russian news corpus we built by crawling a Russian news web site (`www.lenta.ru`). The site provides loose translations of (and pointers to) the original English texts. We collected pairs of articles spanning from 1/1/2001 through 12/24/2004. The corpus consists of 2,022 documents with 0-8 documents per day. The corpus is available on our web page at `http://L2R.cs.uiuc.edu/~cogcomp/`. The English side was tagged with a publicly available NER system based on the SNoW learning architecture (Roth, 1998), that is available at the same site. This set of English NEs was hand-pruned to remove incorrectly classified words to obtain 978 single word NEs.

In order to reduce running time, some limited preprocessing was done on the Russian side. All classes, whose temporal distributions were close to uniform (i.e. words with a similar likelihood of occurrence throughout the corpus) were deemed common and not considered as NE candidates. Unique words were grouped into 15,594 equivalence classes, and 1,605 of those classes were discarded using this method.

Insertions/omissions features were not used in the experiments as they provided no tangible benefit for the languages of our corpus.

Unless mentioned otherwise, the transliteration model was initialized with a subset of 254 pairs of NEs and their transliteration equivalence classes. Negative examples here and during the rest of the training were pairs of randomly selected non-NE English and Russian words.

In each iteration, we used the current transliter-

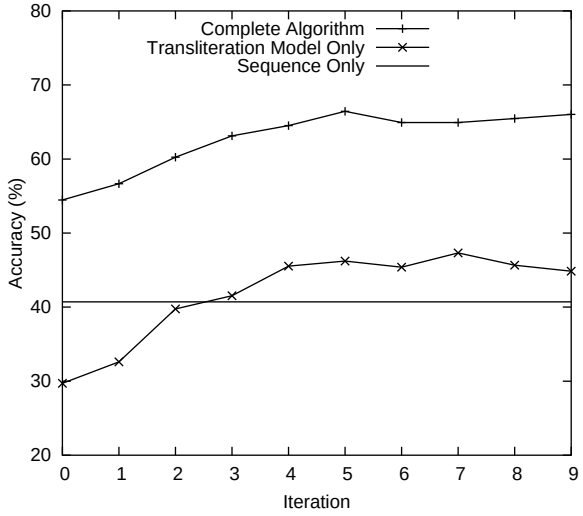


Figure 3: Proportion of correctly discovered NE pairs vs. iteration. Complete algorithm outperforms both transliteration model and temporal sequence matching when used on their own.

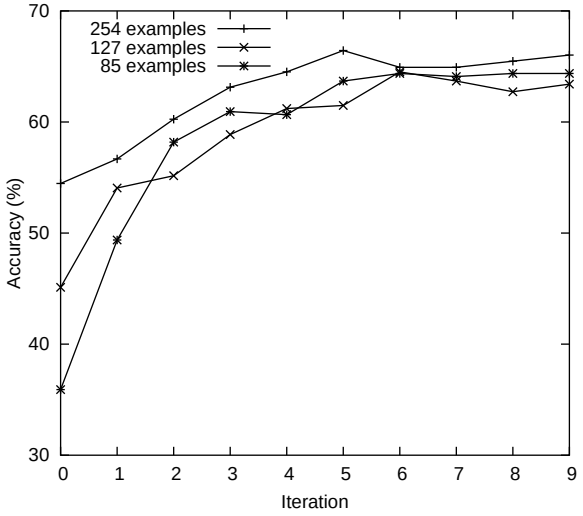


Figure 5: Proportion of the correctly discovered NE pairs for various initial example set sizes. Decreasing the size does not have a significant effect of the performance on later iterations.

ation model to find a list of 30 best transliteration equivalence classes for each NE. We then computed time sequence similarity score between NE and each class from its list to find the one with the best matching time sequence. If its similarity score surpassed a set threshold, it was added to the list of positive examples for the next round of training. Positive examples were constructed by pairing each English NE with each of the transliterations from the best equivalence class that surpasses the threshold. We used the same number of positive and negative examples.

For evaluation, random 727 of the total of 978 NE pairs matched by the algorithm were selected and checked by a language expert. Accuracy was computed as the percentage of those NEs correctly discovered by the algorithm.

4.1 NE Discovery

Figure 3 shows the proportion of correctly discovered NE transliteration equivalence classes throughout the run of the algorithm. The figure also shows the accuracy if transliterations are selected according to the current transliteration model (top scoring candidate) and sequence matching alone. The transliteration model alone achieves an accuracy of about 47%, while the time sequence alone gets about 41%. The combined algorithm achieves about 66%, giving a significant improvement.

In order to understand what happens to the transliteration model as the algorithm proceeds, let us consider the following example. Figure 4 shows parts of transliteration lists for NE *forsyth* for two iterations of the algorithm. The weak transliteration model selects the correct transliteration (italicized) as the 24th best transliteration in the first iteration. Time sequence scoring function chooses it to be one of the training examples for the next round of training of the model. By the eighth iteration, the model has improved to select it as a best transliteration.

Not all correct transliterations make it to the top of the candidates list (transliteration model by itself is never as accurate as the complete algorithm on Figure 3). That is not required, however, as the model only needs to be good enough to place the correct transliteration anywhere in the candidate list.

Not surprisingly, some of the top transliteration candidates start sounding like the NE itself, as training progresses. On Figure 4, candidates for *forsyth* on iteration 7 include *fross* and *fossett*.

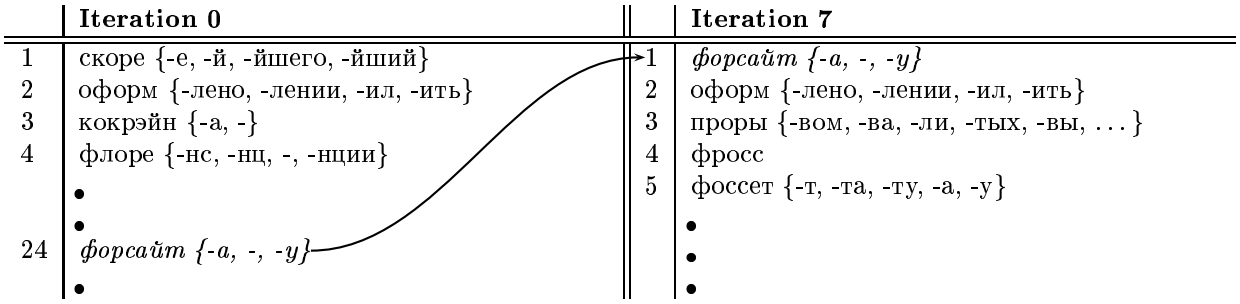


Figure 4: Transliteration lists for *forsyth* for two iterations of the algorithm ranked by the current transliteration model. As the model improves, the correct transliteration moves up the list.

4.2 Rate of Improvement vs. Initial Example Set Size

We ran a series of experiments to see how the size of the initial training set affects the accuracy of the model as training progresses (Figure 5). Although the performance of the early iterations is significantly affected by the size of the initial training example set, the algorithm quickly improves its performance. As we decrease the size from 254, to 127, to 85 examples, the accuracy of the first iteration drops by roughly 10% each time. However, starting at the 6th iteration, the three are with 3% of one another.

These numbers suggest that we only need a few initial positive examples to bootstrap the transliteration model. The intuition is the following: the few examples in the initial training set produce features corresponding to substring pairs characteristic for English-Russian transliterations. Model trained on these (few) examples chooses other transliterations containing these same substring pairs. In turn, the chosen positive examples contain other characteristic substring pairs, which will be used by the model to select more positive examples on the next round, and so on.

English NE	Russian NE equiv. class
lincoln	линкольн {-а, -, -шир}
oregon	орегон {-ского, -е}
niznansky	низнански
uruguay	уругва {-йское, -йцы, -я, -й}
rosing	розингом
gruban	грубан {-ом, -}
meiwes	майвес {-а, -у}
rosetta	розеттского
ecuador	эквадор {-а, -}
laxman	лакшман
friedrich	фридрих {-, -а}
chad	чада

Figure 6: Example of correct transliterations discovered by the algorithm.

5 Conclusions

We have proposed a novel algorithm for cross lingual NE discovery in a bilingual weakly temporally aligned corpus. We have demonstrated that using two independent sources of information (transliteration and temporal similarity) together to guide NE extraction gives better performance than using either of them alone (see Figure 3).

We developed a linear discriminative transliteration model, and presented a method to automatically generate features. For time sequence matching, we used a scoring metric novel in this domain. As supported by our own experiments, this method outperforms other scoring metrics traditionally used (such as *cosine* (Salton and McGill, 1986)) when corpora are not well temporally aligned.

In keeping with our objective to provide as little language knowledge as possible, we introduced a simplistic approach to identifying transliteration equivalence classes, which sometimes produced erroneous groupings (e.g. an equivalence class for NE *lincoln* in Russian included both *lincoln* and *lincolnshire* on Figure 6). This approach is specific to Russian morphology, and would have to be altered for other languages. For example, for Arabic, a small set of prefixes can be used to group most NE variants. We expect that language specific knowl-

edge used to discover accurate equivalence classes would result in performance improvements.

6 Future Work

In this work, we only consider single word Named Entities. A subject of future work is to extend the algorithm to the multi-word setting. Many of the multi-word NEs are translated as well as transliterated. For example, *Mount* in *Mount Rainier* will probably be translated, and *Rainier* - transliterated. If a dictionary exists for the two languages, it can be consulted first, and, if a match is found, transliteration model can be bypassed.

The algorithm can be naturally extended to comparable corpora of more than two languages. Pairwise time sequence scoring and transliteration models should give better confidence in NE matches.

It seems plausible to suppose that phonetic features (if available) would help learning our transliteration model. We would like to verify if this is indeed the case.

The ultimate goal of this work is to automatically tag NEs so that they can be used for training of an NER system for a new language. To this end, we would like to compare the performance of an NER system trained on a corpus tagged using this approach to one trained on a hand-tagged corpus.

7 Acknowledgments

We thank Richard Sproat, ChengXiang Zhai, and Kevin Small for their useful feedback during this work, and the anonymous referees for their helpful comments. This research is supported by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program and a DOI grant under the Reflex program.

References

Nasreen AbdulJaleel and Leah S. Larkey. 2003. Statistical transliteration for english-arabic cross language information retrieval. In *Proceedings of CIKM*, pages 139–146, New York, NY, USA.

George Arfken. 1985. *Mathematical Methods for Physicists*. Academic Press.

Avrim Blum. 1992. Learning boolean functions in an infinite attribute space. *Machine Learning*, 9(4):373–386.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Silviu Cucerzan and David Yarowsky. 1999. Language independent named entity recognition combining morphological and contextual evidence. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Magnus Lie Hetland, 2004. *Data Mining in Time Series Databases*, chapter A Survey of Recent Methods for Efficient Retrieval of Similar Time Sequences. World Scientific.

Sung Young Jung, SungLim Hong, and Eunok Paek. 2000. An english to korean transliteration model of extended markov window. In *Proc. the International Conference on Computational Linguistics (COLING)*, pages 383–389.

Kevin Knight and Jonathan Graehl. 1997. Machine transliteration. In *Proc. of the Meeting of the European Association of Computational Linguistics*, pages 128–135.

Xin Li, Paul Morie, and Dan Roth. 2004. Identification and tracing of ambiguous names: Discriminative and generative approaches. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 419–424.

Robert C. Moore. 2005. A discriminative framework for bilingual word alignment. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 81–88.

Frank Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65.

Dan Roth. 1998. Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 806–813.

Dan Roth. 1999. Learning in natural language. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 898–904.

Gerard Salton and Michael J. McGill. 1986. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA.

Yusuke Shinyama and Satoshi Sekine. 2004. Named entity discovery using comparable news articles. In *Proc. the International Conference on Computational Linguistics (COLING)*, pages 848–853.

Ben Taskar, Simon Lacoste-Julien, and Michael Jordan. 2005. Structured prediction via the extragradient method. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*. MIT Press.

Reducing Weight Undertraining in Structured Discriminative Learning

Charles Sutton, Michael Sindelar, and **Andrew McCallum**
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003 USA
{casutton,mccallum}@cs.umass.edu, msindela@student.umass.edu

Abstract

Discriminative probabilistic models are very popular in NLP because of the latitude they afford in designing features. But training involves complex trade-offs among weights, which can be dangerous: a few highly-indicative features can swamp the contribution of many individually weaker features, causing their weights to be undertrained. Such a model is less robust, for the highly-indicative features may be noisy or missing in the test data. To ameliorate this *weight undertraining*, we introduce several new *feature bagging* methods, in which separate models are trained on subsets of the original features, and combined using a mixture model or a product of experts. These methods include the logarithmic opinion pools used by Smith et al. (2005). We evaluate feature bagging on linear-chain conditional random fields for two natural-language tasks. On both tasks, the feature-bagged CRF performs better than simply training a single CRF on all the features.

1 Introduction

Discriminative methods for training probabilistic models have enjoyed wide popularity in natural language processing, such as in part-of-speech tagging (Toutanova et al., 2003), chunking (Sha and Pereira, 2003), named-entity recognition (Florian et al., 2003; Chieu and Ng, 2003), and most recently parsing (Taskar et al., 2004). A discriminative probabilistic model is trained to maximize the conditional probability $p(\mathbf{y}|\mathbf{x})$ of output labels $\mathbf{y}$ given input variables $\mathbf{x}$, as opposed to modeling the joint probability $p(\mathbf{y}, \mathbf{x})$, as in generative models such as the Naive Bayes classifier and hidden Markov models. The popularity of discriminative models stems from the great flexibility they allow in defining features: because the distribution over input features $p(\mathbf{x})$ is not modeled, it can contain rich, highly overlapping features without making the model intractable for training and inference.

In NLP, for example, useful features include word bigrams and trigrams, prefixes and suffixes, membership in domain-specific lexicons, and information from semantic databases such as WordNet. It is not uncommon to have hundreds of thousands or even millions of features.

But not all features, even ones that are carefully engineered, improve performance. Adding more features to a model can hurt its accuracy on unseen testing data. One well-known reason for this is overfitting: a model with more features has more capacity to fit chance regularities in the training data. In this paper, however, we focus on another, more subtle effect: adding new features can cause existing ones to be *underfit*. Training of discriminative models, such as regularized logistic regression, involves complex trade-offs among weights. A few highly-indicative features can swamp the contribution of many individually weaker features, even if the weaker features, taken together, are just as indicative of the output. Such a model is less robust, for the few strong features may be noisy or missing in the test data.

This effect was memorably observed by Dean Pomerleau (1995) when training neural networks to drive vehicles autonomously. Pomerleau reports one example when the system was learning to drive on a dirt road:

> The network had no problem learning and then driving autonomously in one direction, but when driving the other way, the network was erratic, swerving from one side of the road to the other. ...It turned out that the network was basing most of its predictions on an easily-identifiable ditch, which was always on the right in the training set, but was on the left when the vehicle turned around. (Pomerleau, 1995)

The network had features to detect the sides of the road, and these features were active at training and test time, although weakly, because the dirt road was difficult to

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 89–95,
New York, June 2006.

detect. But the ditch was so highly indicative that the network did not learn the dependence between the road edge and the desired steering direction.

A natural way of avoiding undertraining is to train separate models for groups of competing features—in the driving example, one model with the ditch features, and one with the side-of-the-road features—and then average them into a single model. This is same idea behind *logarithmic opinion pools*, used by Smith, Cohn, and Osborne (2005) to reduce overfitting in CRFs. In this paper, we tailor our ensemble to reduce undertraining rather than overfitting, and we introduce several new combination methods, based on whether the mixture is taken additively or geometrically, and on a per-sequence or per-transition basis. We call this general class of methods *feature bagging*, by analogy to the well-known bagging algorithm for ensemble learning.

We test these methods on conditional random fields (CRFs) (Lafferty et al., 2001; Sutton and McCallum, 2006), which are discriminatively-trained undirected models. On two natural-language tasks, we show that feature bagging performs significantly better than training a single CRF with all available features.

2 Conditional Random Fields

Conditional random fields (CRFs) (Lafferty et al., 2001; Sutton and McCallum, 2006) are undirected graphical models of a conditional distribution. Let $\mathbf{G}$ be an undirected graphical model over random vectors $\mathbf{y}$ and $\mathbf{x}$. As a typical special case, $\mathbf{y} = \{y_t\}$ and $\mathbf{x} = \{x_t\}$ for $t = 1, \ldots, T$, so that $\mathbf{y}$ is a labeling of an observed sequence $\mathbf{x}$. For a given collection $C = \{\{\mathbf{y}_c, \mathbf{x}_c\}\}$ of cliques in $\mathbf{G}$, a CRF models the conditional probability of an assignment to labels $\mathbf{y}$ given the observed variables $\mathbf{x}$ as:

$$p_\Lambda(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in C} \Phi(\mathbf{y}_c, \mathbf{x}_c), \qquad (1)$$

where Φ is a potential function and the partition function $Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{c \in C} \Phi(\mathbf{y}_c, \mathbf{x}_c)$ is a normalization factor over all possible label assignments.

We assume the potentials factorize according to a set of features $\{f_k\}$, which are given and fixed, so that

$$\Phi(\mathbf{y}_c, \mathbf{x}_c) = \exp\left(\sum_k \lambda_k f_k(\mathbf{y}_c, \mathbf{x}_c)\right) \qquad (2)$$

The model parameters are a set of real weights $\Lambda = \{\lambda_k\}$, one weight for each feature.

Many applications have used the *linear-chain CRF*, in which a first-order Markov assumption is made on the hidden variables. In this case, the cliques of the conditional model are the nodes and edges, so that there are feature functions $f_k(y_{t-1}, y_t, \mathbf{x}, t)$ for each label transition. (Here we write the feature functions as potentially depending on the entire input sequence.) Feature functions can be arbitrary. For example, a feature function $f_k(y_{t-1}, y_t, \mathbf{x}, t)$ could be a binary test that has value 1 if and only if y_{t-1} has the label "*adjective*", y_t has the label "*proper noun*", and x_t begins with a capital letter.

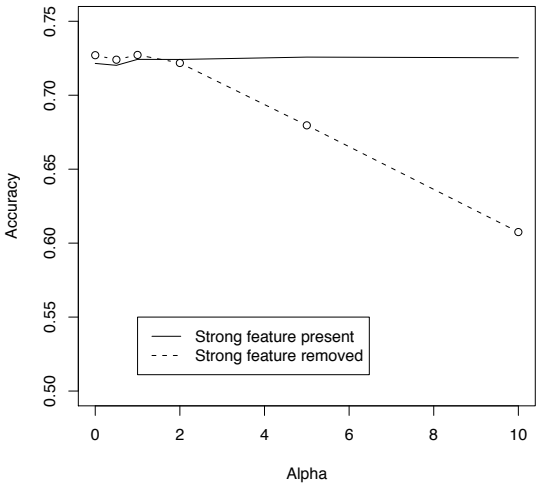


Figure 1: Effect of a single strong feature drowning out weaker features in logistic regression on synthetic data. The x-axis indicates the strength of the strong feature. In the top line, the strong feature is present at training and test time. In the bottom line, the strong feature is missing from the training data at test time.

Linear-chain CRFs correspond to finite state machines, and can be roughly understood as conditionally-trained hidden Markov models (HMMs). This class of CRFs is also a globally-normalized extension to *Maximum Entropy Markov Models* (McCallum et al., 2000) that avoids the label bias problem (Lafferty et al., 2001).

Note that the number of state sequences is exponential in the input sequence length T. In linear-chain CRFs, the partition function $Z(\mathbf{x})$, the node marginals $p(y_i|\mathbf{x})$, and the Viterbi labeling can be calculated efficiently by variants of the dynamic programming algorithms for HMMs.

3 Weight Undertraining

In the section, we give a simple demonstration of weight undertraining. In a discriminative classifier, such as a neural network or logistic regression, a few strong features can drown out the effect of many individually weaker features, even if the weak features are just as indicative put together. To demonstrate this effect, we present an illustrative experiment using logistic regression, because of its strong relation to CRFs. (Linear-

chain conditional random fields are the generalization of logistic regression to sequence data.)

Consider random variables $x_1 \ldots x_n$, each distributed as independent standard normal variables. The output y is a binary variable whose probability depends on all the x_i; specifically, we define its distribution as $y \sim \text{Bernoulli}(\text{logit}(\sum_i x_i))$. The correct decision boundary in this synthetic problem is the hyperplane tangent to the weight vector $(1, 1, \ldots, 1)$. Thus, if n is large, each x_i contributes weakly to the output y. Finally, we include a highly indicative feature $x_S = \alpha \sum_i x_i + \mathcal{N}(\mu = 0, \sigma^2 = 0.04)$. This variable alone is sufficient to determine the distribution of y. The variable α is a parameter of the problem that determines how strongly indicative x_S is; specifically, when $\alpha = 0$, the variable x_S is random noise.

We choose this synthetic model by analogy to Pomerleau's observations. The x_i correspond to the side of the road in Pomerleau's case—the weak features present at both testing and training—and x_S corresponds to the ditch—the strongly indicative feature that is corrupted at test time.

We examine how badly the learned classifier is degraded when x_S feature is present at training time but missing at test time. For several values of the weight parameter α, we train a regularized logistic regression classifier on 1000 instances with $n = 10$ weak variables. In Figure 1, we show how the amount of error caused by ablating x_S at test time varies according to the strength of x_S. Each point in Figure 1 is averaged over 100 randomly-generated data sets. When x_S is weakly indicative, it does not affect the predictions of the model at all, and the classifier's performance is the same whether it appears at test time or not. When x_S becomes strongly indicative, however, the classifier learns to depend on it, and performs much more poorly when x_S is ablated, even though exactly the same information is available in the weak features.

4 Feature Bagging

In this section, we describe the feature bagging method. We divide the set of features $F = \{f_k\}$ into a collection of possibly overlapping subsets $\mathcal{F} = \{F_1, \ldots F_M\}$, which we call *feature bags*. We train individual CRFs on each of the feature bags using standard MAP training, yielding individual CRFs $\{p_1, \ldots p_M\}$.

We average the individual CRFs into a single combined model. This averaging can be performed in several ways: we can average probabilities of entire sequences, or of individual transitions; and we can average using the arithmetic mean, or the geometric mean. This yields four combination methods:

1. *Per-sequence mixture.* The distribution over label sequences $\mathbf{y}$ given inputs $\mathbf{x}$ is modeled as a mixture of the individual CRFs. Given nonnegative weights $\{\alpha_1, \ldots \alpha_m\}$ that sum to 1, the combined model is given by

$$p_{\text{SM}}(\mathbf{y}|\mathbf{x}) = \sum_{i=1}^{M} \alpha_i p_i(\mathbf{y}|\mathbf{x}). \quad (3)$$

It is easily seen that if the sequence model is defined as in Equation 3, then the pairwise marginals are mixtures as well:

$$p_{\text{SM}}(y_t, y_{t-1}|\mathbf{x}) = \sum_{i=1}^{M} \alpha_i p_i(y_t, y_{t-1}|\mathbf{x}). \quad (4)$$

The probabilities $p_i(y_t, y_{t-1}|\mathbf{x})$ are pairwise marginal probabilities in the individual models, which can be efficiently computed by the forward-backward algorithm.

We can perform decoding in the mixture model by maximizing the individual node marginals. That is, to predict y_t we compute

$$y_t^* = \arg\max_{y_t} p_{\text{SM}}(y_t|\mathbf{x}) = \arg\max_{y_t} \sum_i \alpha_i p_i(y_t|\mathbf{x}), \quad (5)$$

where $p_i(y_t|\mathbf{x})$ is computed by first running forward-backward on each of the individual CRFs.

In the results here, however, we compute the maximum probability sequence approximately, as follows. We form a linear-chain distribution $p_{\text{APPX}}(\mathbf{y}|\mathbf{x}) = \prod_t p_{\text{SM}}(y_t|y_{t-1}, \mathbf{x})$, and compute the most probable sequence according to p_{APPX} by the Viterbi algorithm. This is approximate because p_{SM} is not a linear-chain distribution in general, even when all the components are. However, the distribution p_{APPX} does minimize the KL-divergence $D(p_{\text{SM}}\|q)$ over all linear-chain distributions q.

The mixture weights can be selected in a variety of ways, including equal voting, as in traditional bagging, or EM.

2. *Per-sequence product of experts.* These are the logarithmic opinion pools that have been applied to CRFs by (Smith et al., 2005). The distribution over label sequences $\mathbf{y}$ given inputs $\mathbf{x}$ is modeled as a product of experts (Hinton, 2000). In a product of experts, instead of summing the probabilities from the individual models, we multiply them together. Essentially we take a geometric mean instead of an arithmetic mean. Given nonnegative weights $\{\alpha_1, \ldots \alpha_m\}$ that sum to 1, the product model is

$$p(\mathbf{y}|\mathbf{x}) \propto \prod_{i=1}^{M} \left(p_i(\mathbf{y}|\mathbf{x})\right)^{\alpha_i}. \quad (6)$$

The combined model can also be viewed as a conditional random field whose features are the log probabilities from the original models:

$$p(\mathbf{y}|\mathbf{x}) \propto \exp\left\{\sum_{i=1}^{M} \alpha_i \log p_i(\mathbf{y}|\mathbf{x})\right\} \quad (7)$$

By substituting in the CRF definition, it can be seen that the model in Equation 7 is simply a single CRF whose parameters are a weighted average of the original parameters. So feature bagging using the product method does not increase the family of models that are considered: standard training of a single CRF on all available features could potentially pick the same parameters as the bagged model.

Nevertheless, in Section 5, we show that this feature bagging method performs better than standard CRF training.

The previous two combination methods combine the individual models by averaging probabilities of entire sequences. Alternatively, in a sequence model we can average probabilities of individual transitions $p_i(y_t|y_{t-1}, \mathbf{x})$. Computing these transition probabilities requires performing probabilistic inference in each of the original CRFs, because $p_i(y_t|y_{t-1}, \mathbf{x}) = \sum_{\mathbf{y} \setminus y_t, y_{t+1}} p(\mathbf{y}|y_{t-1}, \mathbf{x})$.
This yields two other combination methods:

3. *Per-transition mixture.* The transition probabilities are modeled as

$$p_{\text{TM}}(y_t|y_{t-1}, \mathbf{x}) = \sum_{i=1}^{M} \alpha_i p_i(y_t|y_{t-1}, \mathbf{x}) \quad (8)$$

Intuitively, the difference between per-sequence and per-transition mixtures can be understood generatively. In order to generate a label sequence $\mathbf{y}$ given an input $\mathbf{x}$, the per-sequence model selects a mixture component, and then generates $\mathbf{y}$ using only that component. The per-transition model, on the other hand, selects a component, generates y_1 from that component, selects another component, generates y_2 from the second component given y_1, and so on.

4. *Per-transition product of experts.* Finally, we can combine the transition distributions using a product model

$$p_{\text{SP}}(y_t|y_{t-1}, \mathbf{x}) \propto \prod_{i=1}^{M} p(y_t|y_{t-1}, \mathbf{x})^{\alpha_i} \quad (9)$$

Each transition distribution is thus—similarly to the per-sequence case—an exponential-family distribution whose features are the log transition probabilities from the individual models. Unlike the per-sequence product, there is no weight-averaging trick here, because the probabilities $p(y_t|y_{t-1}, \mathbf{x})$ are marginal probabilities.

Considered as a sequence distribution $p(\mathbf{y}|\mathbf{x})$, the per-transition product is a locally-normalized maximum-entropy Markov model (McCallum et al., 2000). It would not be expected to suffer from label bias, however, because each of the features take the future into account; they are marginal probabilities from CRFs.

Of these four combination methods, Method 2, the per-sequence product of experts, is originally due to Smith et al. (2005). The other three combination methods are as far as we know novel. In the next section, we compare the four combination methods on several sequence labeling tasks. Although for concreteness we describe them in terms of sequence models, they may be generalized to arbitrary graphical structures.

5 Results

We evaluate feature bagging on two natural language tasks, named entity recognition and noun-phrase chunking. We use the standard CoNLL 2003 English data set, which is taken from Reuters newswire and consists of a training set of 14987 sentences, a development set of 3466 sentences, and a testing set of 3684 sentences. The named-entity labels in this data set corresponding to people, locations, organizations and other miscellaneous entities. Our second task is noun-phrase chunking. We use the standard CoNLL 2000 data set, which consists of 8936 sentences for training and 2012 sentences for testing, taken from Wall Street Journal articles annotated by the Penn Treebank project. Although the CoNLL 2000 data set is labeled with other chunk types as well, here we use only the NP chunks.

As is standard, we compute precision and recall for both tasks based upon the chunks (or named entities for CoNLL 2003) as

$$P = \frac{\text{\# correctly labeled chunks}}{\text{\# labeled chunks}}$$

$$R = \frac{\text{\# correctly labeled chunks}}{\text{\# actual chunks}}$$

We report the harmonic mean of precision and recall as $F_1 = (2PR)/(P + R)$.

For both tasks, we use per-sequence product-of-experts feature bagging with two feature bags which we manually choose based on prior experience with the data set. For each experiment, we report two baseline CRFs, one trained on union of the two feature sets, and one trained only on the features that were present in both bags, such as lexical identity and regular expressions. In both data

sets, we trained the individual CRFs with a Gaussian prior on parameters with variance $\sigma^2 = 10$.

For the named entity task, we use two feature bags based upon character ngrams and lexicons. Both bags contain a set of baseline features, such as word identity and regular expressions (Table 4). The ngram CRF includes binary features for character ngrams of length 2, 3, and 4 and word prefixes and suffixes of length 2, 3, and 4. The lexicon CRF includes membership features for a variety of lexicons containing people names, places, and company names. The combined model has 2,342,543 features. The mixture weight α is selected using the development set.

For the chunking task, the two feature sets are selected based upon part of speech and lexicons. Again, a set of baseline features are used, similar to the regular expressions and word identity features used on the named entity task (Table 4). The first bag also includes part-of-speech tags generated by the Brill tagger and the conjunctions of those tags used by Sha and Pereira (2003). The second bag uses lexicon membership features for lexicons containing names of people, places, and organizations. In addition, we use part-of-speech lexicons generated from the entire Treebank, such as a list of all words that appear as nouns. These lists are also used by the Brill tagger (Brill, 1994). The combined model uses 536,203 features. The mixture weight α is selected using 2-fold cross validation. The chosen model had weight 0.55 on the lexicon model, and weight 0.45 on the ngram model.

In both data sets, the bagged model performs better than the single CRF trained with all of the features. For the named entity task, bagging improves performance from 85.45% to 86.61%, with a substantial error reduction of 8.32%. This is lower than the best reported results for this data set, which is 89.3% (Ando and Zhang, 2005), using a large amount of unlabeled data. For the chunking task, bagging improved the performance from 94.34% to 94.77%, with an error reduction of 7.60%. In both data sets, the improvement is statistically significant (McNemar's test; $p < 0.01$).

On the chunking task, the bagged model also outperforms the models of Kudo and Matsumoto (2001) and Sha and Pereira (2003), and equals the currently-best results of (Ando and Zhang, 2005), who use a large amount of unlabeled data. Although we use lexicons that were not included in the previous models, the additional features actually do not help the original CRF. Only with feature bagging do these lexicons improve performance.

Finally, we compare the four bagging methods of Section 4: pre-transition mixture, pre-transition product of experts, and per-sequence mixture. On the named entity data, all four models perform in a statistical tie, with no statistically significant difference in their performance (Table 1). As we mentioned in the last section, the decoding procedure for the per-sequence mixture is approximate. It is possible that a different decoding procedure, such as maximizing the node marginals, would yield better performance.

Model	F1
Per-sequence Product of Experts	86.61
Per-transition Product of Experts	86.58
Per-sequence Mixture	86.46
Per-transition Mixture	86.42

Table 1: Comparison of various bagging methods on the CoNLL 2003 Named Entity Task.

Model	F1
Single CRF(Base Feat.)	81.52
Single CRF(All Feat.)	85.45
Combined CRF	86.61

Table 2: Results for the CoNLL 2003 Named Entity Task. The bagged CRF performs significantly better than a single CRF with all available features (McNemar's test; $p < 0.01$).

6 Previous Work

In the machine learning literature, there is much work on ensemble methods such as stacking, boosting, and bagging. Generally, the ensemble of classifiers is generated by training on different subsets of data, rather than different features. However, there is some literature within unstructured classified on combining models trained on feature subsets. Ho (1995) creates an ensemble of decision trees by randomly choosing a feature subset on which to grow each tree using standard decision tree learners. Other work along these lines include that of Bay (1998) using nearest-neighbor classifiers, and more recently Bryll et al (2003). Also, in Breiman's work on random forests (2001), ensembles of random decision trees are constructed by choosing a random feature at each node. This literature mostly has the goal of improving accuracy by reducing the classifier's variance, that is, reducing overfitting.

In contrast, O'Sullivan et al. (2000) specifically focus on increasing robustness by training classifiers to use all of the available features. Their algorithm FeatureBoost is analogous to AdaBoost, except that the meta-learning algorithm maintains weights on features instead of on instances. Feature subsets are automatically sampled based on which features, if corrupted, would most affect the ensemble's prediction. They show that FeatureBoost is more robust than AdaBoost on synthetically corrupted UCI data sets. Their method does not easily extend to sequence models, especially natural-language models with hundreds of thousands of features.

Model	F1
Single CRF(Base Feat.)	89.60
Single CRF(All Feat.)	94.34
(Sha and Pereira, 2003)	94.38
(Kudo and Matsumoto, 2001)	94.39
(Ando and Zhang, 2005)	94.70
Combined CRF	94.77

Table 3: Results for the CoNLL 2000 Chunking Task. The bagged CRF performs significantly better than a single CRF (McNemar's test; $p < 0.01$), and equals the results of (Ando and Zhang, 2005), who use a large amount of unlabeled data.

$w_t = w$
w_t begins with a capital letter
w_t contains only capital letters
w_t is a single capital letter
w_t contains some capital letters and some lowercase
w_t contains a numeric character
w_t contains only numeric characters
w_t appears to be a number
w_t is a string of at least two periods
w_t ends with a period
w_t contains a dash
w_t appears to be an acronym
w_t appears to be an initial
w_t is a single letter
w_t contains punctuation
w_t contains quotation marks
$P_t = P$
All features for time $t + \delta$ for all $\delta \in [-2, 2]$

Table 4: Baseline features used in all bags. In the above w_t is the word at position t, P_t is the POS tag at position t, w ranges over all words in the training data, and P ranges over all chunk tags supplied in the training data. The "appears to be" features are based on hand-designed regular expressions.

There is less work on ensembles of sequence models, as opposed to unstructured classifiers. One example is Altun, Hofmann, and Johnson (2003), who describe a boosting algorithm for sequence models, but they boost instances, not features. In fact, the main advantage of their technique is increased model sparseness, whereas in this work we aim to fully use *more* features to increase accuracy and robustness.

Most closely related to the present work is that on logarithmic opinion pools for CRFs (Smith et al., 2005), which we have called per-sequence mixture of experts in this paper. The previous work focuses on reducing overfitting, combining a model of many features with several simpler models. In contrast, here we apply feature bagging to reduce feature undertraining, combining several models with complementary feature sets. Our current positive results are probably not due to reduction in overfitting, for as we have observed, all the models we test, including the bagged one, have 99.9% F1 on the training set. Now, feature undertraining can be viewed as a type of overfitting, because it arises when a set of features is more indicative in the training set than the testing set. Understanding this particular type of overfitting is useful, because it motivates the choice of feature bags that we explore in this work. Indeed, one contribution of the present work is demonstrating how a careful choice of feature bags can yield state-of-the-art performance.

Concurrently and independently, Smith and Osborne (2006) present similar experiments on the CoNLL-2003 data set, examining a per-sequence mixture of experts (that is, a logarithmic opinion pool), in which the lexicon features are trained separately. Their work presents more detailed error analysis than we do here, while we present results both on other combination methods and on NP chunking.

7 Conclusion

Discriminatively-trained probabilistic models have had much success in applications because of their flexibility in defining features, but sometimes even highly-indicative features can fail to increase performance. We have shown that this can be due to feature undertraining, where highly-indicative features prevent training of many weaker features. One solution to this is feature bagging: repeatedly selecting feature subsets, training separate models on each subset, and averaging the individual models.

On large, real-world natural-language processing tasks, feature bagging significantly improves performance, even with only two feature subsets. In this work, we choose the subsets based on our intuition of which features are complementary for this task, but automatically determining the feature subsets is an interesting area for future work.

Acknowledgments

We thank Andrew Ng, Hanna Wallach, Jerod Weinman, and Max Welling for helpful conversations. This work was supported in part by the Center for Intelligent Information Retrieval, in part by the Defense Advanced Research Projects Agency (DARPA), in part by The Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grant #IIS-0326249, and in part by The Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grant #IIS-0427594. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

References

Yasemin Altun, Thomas Hofmann, and Mark Johnson. 2003. Discriminative learning for label sequences via boosting. In *Advances in Neural Information Processing Systems (NIPS*15)*.

Rie Ando and Tong Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 1–9, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Stephen D. Bay. 1998. Combining nearest neighbor classifiers through multiple feature subsets. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 37–45. Morgan Kaufmann Publishers Inc.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32, October.

Eric Brill. 1994. Some advances in transformation-based part of speech tagging. In *AAAI '94: Proceedings of the twelfth national conference on Artificial intelligence (vol. 1)*, pages 722–727. American Association for Artificial Intelligence.

Robert Bryll, Ricardo Gutierrez-Osuna, and Francis Quek. 2003. Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. *Pattern Recognition*, 36:1291–1302.

Hai Leong Chieu and Hwee Tou Ng. 2003. Named entity recognition with a maximum entropy approach. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 160–163. Edmonton, Canada.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of CoNLL-2003*.

G.E. Hinton. 2000. Training products of experts by minimizing contrastive divergence. Technical Report 2000-004, Gatsby Computational Neuroscience Unit.

T. K. Ho. 1995. Random decision forests. In *Proc. of the 3rd Int'l Conference on Document Analysis and Recognition*, pages 278–282, Montreal, Canada, August.

T. Kudo and Y. Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of NAACL-2001*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proc. 18th International Conf. on Machine Learning*.

Andrew McCallum, Dayne Freitag, and Fernando Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proc. 17th International Conf. on Machine Learning*, pages 591–598. Morgan Kaufmann, San Francisco, CA.

Joseph O'Sullivan, John Langford, Rich Caruana, and Avrim Blum. 2000. Featureboost: A meta learning algorithm that improves model robustness. In *International Conference on Machine Learning*.

Dean Pomerleau. 1995. Neural network vision for robot driving. In M. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL 2003*. Association for Computational Linguistics.

Andrew Smith and Miles Osborne. 2006. Using gazetteers in discriminative information extraction. In *CoNLL-X, Tenth Conference on Computational Natural Language Learning*.

Andrew Smith, Trevor Cohn, and Miles Osborne. 2005. Logarithmic opinion pools for conditional random fields. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 18–25, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Charles Sutton and Andrew McCallum. 2006. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press. To appear.

Ben Taskar, Dan Klein, Michael Collins, Daphne Koller, and Chris Manning. 2004. Max-margin parsing. In *Empirical Methods in Natural Language Processing (EMNLP04)*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL 2003*.

A Maximum Entropy Approach to Combining Word Alignments

Necip Fazil Ayan and **Bonnie J. Dorr**
Institute of Advanced Computer Studies (UMIACS)
University of Maryland
College Park, MD 20742
{nfa,bonnie}@umiacs.umd.edu

Abstract

This paper presents a new approach to combining outputs of existing word alignment systems. Each alignment link is represented with a set of feature functions extracted from linguistic features and input alignments. These features are used as the basis of alignment decisions made by a maximum entropy approach. The learning method has been evaluated on three language pairs, yielding significant improvements over input alignments and three heuristic combination methods. The impact of word alignment on MT quality is investigated, using a phrase-based MT system.

1 Introduction

Word alignment—detection of corresponding words between two sentences that are translations of each other—is usually an intermediate step of statistical machine translation (MT) (Brown et al., 1993; Och and Ney, 2003; Koehn et al., 2003), but also has been shown useful for other applications such as construction of bilingual lexicons, word-sense disambiguation, projection of resources, and cross-language information retrieval.

Maximum entropy (ME) models have been used in bilingual sense disambiguation, word reordering, and sentence segmentation (Berger et al., 1996), parsing, POS tagging and PP attachment (Ratnaparkhi, 1998), machine translation (Och and Ney, 2002), and FrameNet classification (Fleischman et al., 2003). They have also been used to solve the word alignment problem (Garcia-Varea et al., 2002; Ittycheriah and Roukos, 2005; Liu et al., 2005), but a sentence-level approach to combining knowledge sources is used rather than a word-level approach.

This paper describes an approach to combining evidence from alignments generated by existing systems to obtain an alignment that is closer to the true alignment than the individual alignments. The alignment-combination approach (called *ACME*) operates at the level of alignment links, rather than at the sentence level (as in previous ME approaches). ACME uses ME to decide whether to include/exclude a particular alignment link based on feature functions that are extracted from the input alignments and linguistic features of the words. Since alignment combination relies on evidence from existing alignments, we focus on alignment links that exist in at least one input alignment. An important challenge in this approach is the selection of appropriate links when two aligners make different alignment choices.

We show that ACME yields a significant relative error reduction over the input alignment systems and heuristic-based combinations on three different language pairs. Using a higher number of input alignments and partitioning the training data into disjoint subsets yield further error-rate reductions.

The next section briefly overviews ME models. Section 3 presents a new ME approach to combining existing word alignment systems. Section 4 describes the evaluation data, input alignments, and evaluation metrics. Section 5 presents experiments on three language pairs, upper bounds for alignment error rate in alignment combination, and MT evaluation on English-Chinese and English-Arabic. Section 6 describes previous work on alignment combination and ME models on word alignment.

2 Maximum Entropy (ME) Models

In a statistical classification problem, the goal is to estimate the probability of a class y in a given context x, i.e., $p(y|x)$. In an ideal scenario, if the training data contain evidence for all pairs of (y, x), it is

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 96–103,
New York, June 2006.

trivial to compute the probability distribution p. Unfortunately, due to training-data sparsity, p is generally modeled using only the available evidence.

Given a collection of facts, ME chooses a model consistent with all the facts, but otherwise as uniform as possible (Berger et al., 1996). Formally, the evidence is represented as feature functions, i.e., binary valued functions that map a class y and a context x to either 0 or 1, i.e., $h_m : \mathcal{Y} \times \mathcal{X} \rightarrow \{0, 1\}$, where $\mathcal{Y}$ is the set of all classes and $\mathcal{X}$ is the set of all facts. The biggest advantage of maximum entropy models is that they are able to focus on the selection of feature functions rather than on how such functions are used. Any context can be used to define feature functions without concern for the independence of the feature functions from each other or the relevance of the feature functions to the final decision (Ratnaparkhi, 1998).

Each feature function h_m is associated with a model parameter λ_m. Given a set of M feature functions $h_1, \ldots, h_M$, the probability of class y given a context x is equal to:

$$p(y|x) = \frac{1}{Z_x} exp\left(\sum_{m=1}^{M} \lambda_m h_m(y, x)\right)$$

where Z_x is a normalization constant. The contribution of each feature function to the final decision, i.e., λ_m, can be automatically computed using Generalized Iterative Scaling (GIS) algorithm (Darroch and Ratcliff, 1972). The final classification for a given instance is the class y that maximizes $p(y|x)$.

3 Alignment Combination: ACME

Let $\mathbf{e} = e_1, \ldots, e_I$ and $\mathbf{f} = f_1, \ldots, f_J$ be two sentences in two different languages. An alignment link (i, j) corresponds to a translational equivalence between words e_i and f_j. Let A_k be an alignment between sentences $\mathbf{e}$ and $\mathbf{f}$, where each element $a \in A_k$ is an alignment link (i, j). Let $\mathcal{A} = \{A_1, \ldots, A_n\}$ be a set of alignments between $\mathbf{e}$ and $\mathbf{f}$. We refer to the true alignment as T, where each $a \in T$ is of the form (i, j). The goal of ACME is to combine the information in $\mathcal{A}$ such that the combined alignment A_C is closer to T. A straightforward solution is to take the intersection or union of the individual alignments. In this paper, an additional model is learned to combine outputs of $A_1, \ldots, A_n$.

In our combination framework, first, n different word-alignment systems, $A_1, \ldots, A_n$, generate word alignments between a given English sentence and a foreign-language (FL) sentence. Then a *Feature Extractor* takes the output of these alignment systems and the parallel corpus (which might be enriched with linguistic features) and extracts a set of feature functions based on linguistic properties of the words and the input alignments. Each feature function h_m is associated with a model parameter λ_m. Next, an *Alignment Combiner* decides whether to include or exclude an alignment link based on the extracted feature functions and the model parameters associated with them.

For each possible alignment link a set of features is extracted from the input alignments and linguistic properties of words. The features that are used for representing an alignment link (i, j) are as follows:

1. **Part-of-speech tags (*posE, posF, prevposE, prevposF, nextpostE, nextposF*):** POS tags for the previous, current, and the next English and FL words.
2. **Outputs of input aligners (*out*):** Whether (i, j) exists in a given input alignment A_k.
3. **Neighbors (*neigh*):** A *neighborhood* of an alignment link (i, j)—denoted by $N(i, j)$—consists of 8 possible alignment links in a 3×3 window with (i, j) in the center of the window. Each element of $N(i, j)$ is called a *neighboring link* of (i, j). Neighbor features include: (1) Whether a particular neighbor of (i, j) exists in a given input alignment A_k; and (2) Total number of neighbors of (i, j) in a given input alignment A_k.
4. **Fertilities (*fertE, fertF*):** The number of words that e_i (or f_j) is aligned to in a given input alignment A_k.
5. **Monotonicity (*mon*):** The absolute difference between i and j.

Our combination approach employs feature functions derived from a subset of the features above. Assuming $\mathcal{Y} = \{\text{yes}, \text{no}\}$ represents the set of classes, where each class denotes the existence or absence of a link in the combined alignment, and $\mathcal{X}$ is the set of features above, we generate various feature functions $h(y, x)$, where $y \in \mathcal{Y}$ and x are instantiations of one or more features in $\mathcal{X}$. Table 1 lists the feature sets with an example feature func-

Features	Example Feature Function	
$posE$	$h('\text{yes}', i, j) = 1$	if $(i, j) \in A_C$ and $pos(e_i) = Noun$
$posF$	$h('\text{no}', i, j) = 1$	if $(i, j) \notin A_C$ and $pos(f_j) = Verb$
out	$h('\text{yes}', i, j, k) = 1$	if $(i, j) \in A_C$ and $(i, j) \in A_k$
$out, neigh$	$h('\text{yes}', i, j, k) = 1$	if $(i, j) \in A_C$ and $(i-1, j+1) \in A_k$
	$h('\text{yes}', i, j, k) = 1$	if $(i, j) \in A_C$ and $\|NC\| = 2$ where $NC = \{n \| n \in N(i, j), n \in A_k\}$
$out, fertE$	$h('\text{no}', i, j, k) = 1$	if $(i, j) \notin A_C$ and $\|FT\| = 0$ where $FT = \{t \| (i, t) \in A_k\}$
$out, fertF$	$h('\text{no}', i, j, k) = 1$	if $(i, j) \notin A_C$ and $\|FT\| = 1$ where $FT = \{t \| (t, j) \in A_k\}$
mon	$h('\text{yes}', i, j) = 1$	if $(i, j) \in A_C$ and $\|i - j\| = 2$

Table 1: Feature Functions.

tion for each.[1] For example, the feature function in the fifth row has a value of 1 if there are 2 neighboring links to (i, j) that exist in the input alignment A_k and the alignment link (i, j) exists in A_C.

In combining evidence from different alignments, it is assumed that, when an alignment link is left out by all aligners, that particular link should not be included in the final output. Since the majority of all possible word pairs are unaligned in real data, the inclusion of all possible word pairs in the training data leads to skewed results, where the learning algorithm is biased toward labeling the links as invalid. To offset this problem, our training data includes only alignment links that appear in at least one input alignment.

Once the feature functions are extracted, we learn the model parameters using the YASMET ME package (Och, 2002), which is an efficient implementation of the GIS algorithm.

4 Experiment Data, Alignment Inputs, and Metrics

The alignment combination techniques are evaluated in this paper using data from three language pairs, as shown in Table 2.

Lang Pair	# of Sent's	# Words (en/fl)	Source
en-ch	491	13K/13K	NIST MTEval '02[2]
en-ar	450	11K/13K	NIST MTEval '03[3]
en-ro	248	5.5K/5.5K	HLT Workshop '03[4]

Table 2: Data Used for Combination Experiments.

Input alignments are generated using two existing word alignment systems: GIZA++ (Och, 2000) and SAHMM (Lopez and Resnik, 2005). Both systems are run in two different directions with default configurations. We indicate the two directions using the notation $Aligner(en \rightarrow fl)$ and $Aligner(fl \rightarrow en)$, where en is English, fl is either Chinese (*ch*), Arabic (*ar*), or Romanian (*ro*).

To train both systems, additional data was used for the three language pairs: 107K English-Chinese sentence pairs (4.1M/3.3M English/Chinese words); 44K English-Arabic sentence pairs (1.4M/1M English/Arabic words); 48K English-Romanian sentence pairs (1M/1M English/Romanian words).[5]

POS tags were generated using the MXPOST tagger (Ratnaparkhi, 1998). POS tagger for English was trained on Sections 0-18 of the Penn Treebank Wall Street Journal corpus. On the FL side, we used POS tagger for only Chinese and it was trained on Sections 16-299 of Chinese Treebank.

For comparison purposes, three additional heuristically-induced alignments are generated for each system: (1) Intersection of both directions ($Aligner$(int)); (2) Union of both directions ($Aligner$(union)); and (3) The previously best-known heuristic combination approach called *grow-diag-final* (Koehn et al., 2003) ($Aligner$(gdf)).

In our evaluation, we take A to be the set of alignment links for a set of sentences, S to be the set of sure alignment links, and P be the set of probable alignment links (in the gold standard). Precision (Pr), recall (Rc) and alignment error rate (AER) are defined as follows:[6]

$$Pr = \frac{|A \cap P|}{|A|} \quad Rc = \frac{|A \cap S|}{|S|}$$

$$AER = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}$$

[1] In Table 1, *NC* corresponds to the set of (i, j)'s neighbors that exist in the alignment A_k, and *FT* represents the set of words that e_i (or f_j) is aligned to.

[2] From (Ayan et al., 2005).

[3] From (Ittycheriah and Roukos, 2005).

[4] From (Mihalcea and Pedersen, 2003).

[5] Note that both GIZA++ and SAHMM are unsupervised learning systems. Sentence-aligned parallel texts are the only required input.

[6] Note that AER= 1 - F-score when there is no distinction between probable and sure alignment links.

Our gold standard for each language pair is a manually aligned corpus. English-Chinese annotations distinguish between sure and probable alignment links (i.e., $S \subset P$), but there is no such distinction for the other two language pairs (i.e., $P = S$).

Because of the availability of limited manually annotated data, evaluations are performed using 5-fold cross validation. Once the alignments are generated for each fold (using one as the test set and the other 4 folds as training set), the results are concatenated to compute precision, recall and error rate on the entire set of sentence pairs for each data set.[7]

5 Experiments and Results

This section presents several experiments and results comparing AER of ACME to those of standard alignment approaches on English-Chinese data. We also present experiments on additional languages, analyses based on precision and recall, an upper-bound oracle analysis, and MT evaluations.

5.1 English-Chinese Experiments

The experiments below test the effects of input alignments, feature set, data partitioning, number of inputs, and size of training data on the performance of ACME.

2 Input alignments: Table 3 shows the AER for GIZA++ and SAHMM (in each direction), three heuristic-based combinations and ACME using 2 uni-directional alignments as input and all features described in Section 3.[8] (We use 'ACME[2]' in this section to refer to ACME applied to two input alignments and ACME[4] in later sections to refer to ACME applied to four input alignments.)

Using 2 GIZA++ uni-directional alignments as input, ACME yields a 22.0% AER—a relative error reduction of 25.9% over GIZA++(gdf). Similarly, using 2 SAHMM uni-directional alignments as input, ACME produces a 20.6% AER—a relative error reduction of 28.0% and 25.4% over SAHMM(gdf) and SAHMM(int), respectively.

[7] Because the NIST MTEval data include sentences that may be related (according to the document in which they appear), the training and test material could potentially be related; however, given the types of features used in our experiments, we do not believe this biases our results.

[8] For ease of readability, in the rest of this paper, we will report precision, recall, and AER in percentages.

Alignments	GIZA++	SAHMM
$Aligner(en \rightarrow fl)$	30.7	26.5
$Aligner(fl \rightarrow en)$	32.2	31.3
$Aligner(int)$	31.2	27.6
$Aligner(union)$	31.6	29.8
$Aligner(gdf)$	29.7	28.6
ACME[2]	22.0	20.6

Table 3: Comparison of GIZA++ and SAHMM to ACME[2] (on English-Chinese).

Feature Set: To examine the effects of each feature on the performance of ACME, we compute the AER under a variety of conditions, removing each feature one at a time. ACME is evaluated using 2 uni-directional GIZA++ alignments as input on English-Chinese data. Using all features, the AER is 22.0%. Our experiments show that there is no significant increase in AER for the removal of features corresponding to monotonicity (22.1%), neighbors (22.8%), POS on English side (22.9%), POS on foreign-language side (22.9%). On the other hand, deleting POS tags on both sides yields an AER of 25.2% and deleting the fertility features increases the AER to 25.9%. This indicates that both POS tags (or fertilities) contribute heavily toward the decision as to whether a particular alignment should be included/excluded.

Partitioning Data: Previous work showed that partitioning the data into disjoint subsets and learning a different model for each partition improves the performance of the alignment systems (Ayan et al., 2005). To test whether this same principle applies to alignment combination with maximum entropy modeling, the training data was partitioned using POS tags for English and the FL, and different weights were learned for each partition.

Alignments	GIZA++	SAHMM
ACME[2]	22.0	20.6
ACME[2]-Part[$posE$]	19.8	18.0
ACME[2]-Part[$posF$]	20.0	18.1
ACME[2]-Part[$posE, posF$]	20.0	18.4

Table 4: Application of ACME[2] on Partitioned Data (on English-Chinese).

Table 4 presents the AER for ACME[2], using either two GIZA++ alignments or two SAHMM alignments, on English-Chinese data. Without any partitioning, ACME achieves an AER of 22.0 (GIZA++) and 20.6 (SAHMM). Using English POS tags for data partitioning results in a significant reduction

in AER: 19.8% (GIZA++) and 18.0% (SAHMM). Interestingly, using foreign-language (FL) tags on their own or together with English POS tags does not provide any improvement. Overall when ACME[2] is applied to partitioned data (using *posE* for partitioning) a relative error reduction of 33–37% over GIZA++(gdf) and SAHMM(gdf) is achieved.

Number of Input Alignments: Table 5 presents the English-Chinese AER for ACME[1] (using either GIZA++ or SAHMM in only one direction), ACME[2] (using either GIZA++ or SAHMM in two directions) and ACME[4] (using GIZA++ and SAHMM, each in two directions).

Regardless of the number of inputs, partitioning the data (using English POS tags) yields lower AER than no partitioning. Using one GIZA++ alignment as input, ACME[1] with partitioning improves the AER to 26.9% and 25.5% for each direction, respectively. Similarly, using one SAHMM alignment as input, ACME[1] with partitioning reduces the AER to 22.9% and 24.7%. ACME[2] with partitioning reduces the AER to 19.8% and 18.0% for GIZA++ and SAHMM, respectively. Finally, using all four input alignments, ACME[4] with partitioning yields a 15.6% AER—a relative error reduction of 21.2% and 13.3% over each ACME[2] case.

Alignments	GIZA++	SAHMM
ACME[1]($en \rightarrow fl$)	28.1	24.4
ACME[1]-Part[posE]($en \rightarrow fl$)	26.9	22.9
ACME[1]($fl \rightarrow en$)	26.6	26.9
ACME[1]-Part[posE]($fl \rightarrow en$)	25.5	24.7
ACME[2]	22.0	20.6
ACME[2]-Part[posE]	19.8	18.0
ACME[4]	17.8	
ACME[4]-Part[posE]	15.6	

Table 5: Application of ACME to 1, 2 and 4 Input Alignments (on English-Chinese).

Size of Training Data to Obtain Input Alignments: In general, statistical alignment systems improve as the size of the training data increases. We present the AER for GIZA++ and ACME[2] using GIZA++ alignments as input, where GIZA++ is trained on different sizes of data. We started with 20K sentence pairs of FBIS data and increased it to all available FBIS data (241K sentence pairs).

Figure 1 compares the alignment performance of: (1) uni-directional GIZA++ (each direction); (2) GIZA++(gdf); and (3) ACME[2] with all features and English POS partitioning. With only 20K sentence pairs, ACME[2] achieves an AER of 23.7% in contrast to 34.3% AER for GIZA++(gdf). With 241K sentence pairs, ACME[2] yields 18.3% AER in contrast to 27.7% AER for GIZA++(gdf). We should emphasize that ACME[2] on only 20K sentence pairs yields a lower AER than those of all GIZA++ alignments obtained on 241K sentence pairs. Overall ACME[2] achieves a relative error reduction of 31–38% over the input alignments, and a relative error reduction of 31–34% over GIZA++(gdf) for different sizes of training data.

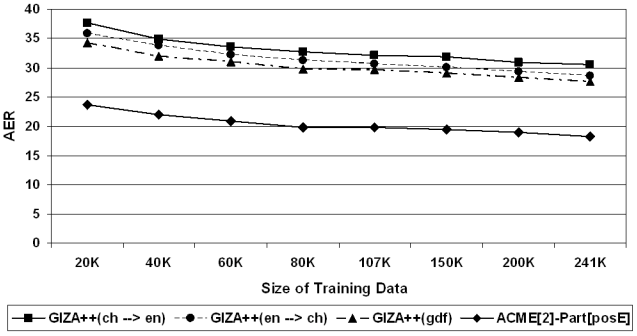


Figure 1: Effects of Training Data Size Used for Initial Alignments on the performance of GIZA++ and ACME[2] (on English-Chinese).

5.2 Expanding to Additional Languages

We also investigated the applicability of ACME to additional language pairs. Table 6 presents the AER for GIZA++ and SAHMM (in each direction), three combination heuristics (gdf, int and union), and ACME[2] and ACME[4] on English-Arabic and English-Romanian data. We should emphasize that no POS tagger on the FL side was used for these experiments.

On English-Arabic data, ACME[2] (with POS partitioning and including all features) yields 21.4% (20.7%) AER—a relative error reduction of 24.6% (13.0%) over the best combination heuristic with GIZA++ (SAHMM) alignments. ACME[4] reduces the AER to 18.1%—a relative error reduction of 36.3% and 23.9% over GIZA++(int) and SAHMM(int), respectively.

On English-Romanian data, ACME[2] (with POS partitioning and including all features) yields 24.7% (26.2%) AER—a relative error reduction of 14.3% (10.6%) over the best combination heuristic with GIZA++ (SAHMM) alignments. ACME[4] re-

	English-Arabic		English-Romanian	
Alignments	**GIZA++**	**SAHMM**	**GIZA++**	**SAHMM**
$Aligner(en \rightarrow fl)$	34.5	27.8	32.7	31.0
$Aligner(fl \rightarrow en)$	27.9	29.5	30.0	29.8
$Aligner(int)$	28.4	23.8	32.7	29.3
$Aligner(union)$	32.8	32.0	30.5	31.2
$Aligner(gdf)$	30.2	30.4	28.8	30.3
ACME[2]	23.2	21.9	25.2	27.0
ACME[2]-Part[$posE$]	21.4	20.7	24.7	26.2
ACME[4]	19.8		24.0	
ACME[4]-Part[$posE$]	18.1		22.3	

Table 6: AER for Input Alignments, Heuristic-based Alignments, and ACME Using 2 and 4 Input Alignments (on English-Arabic and English-Romanian).

duces the AER to 22.3%—a relative error reduction of 22.6% and 23.9% over GIZA++(int) and SAHMM(int), respectively.

5.3 Precision, Recall and Upper-Bound Analysis

We now turn to a precision vs. recall analysis of different alignments to elucidate the nature of the differences between two alignments.

Figure 2 presents precision and recall values for three combined alignments using GIZA++ (int, union, gdf) as well as results for ACME[2] and ACME[4] on three different language pairs. For all three pairs, the ranking of the combined alignments is the same with respect to precision and recall. GIZA++(int) yields the highest precision (nearly 95%) but the lowest recall (53–57%). Both union and gdf methods achieve low precision (56–68%) but high recall (75–83%), and gdf is better than union. By contrast, ACME[2] yields significantly higher precision (nearly 87%) but lower recall (67–75%) with respect to union and gdf. ACME[4] has higher precision and recall than ACME[2]—an absolute increase of 2–3% and 4%, respectively.

Next we compute an oracle upper-bound in AER where mismatched input alignments are assumed to be resolved perfectly within the alignment combination framework (i.e., an *oracle* chooses the correct output in cases where the input aligners make different choices).[9]

Table 7 presents the upper bounds using a generic alignment combiner (denoted *Oracle*) with 2 and 4 input alignments on three language pairs, assuming a perfect resolution of mismatched input alignments. For English-Chinese, the upper bound is 9.4% (using *Oracle*[2]) and 4.7% (using *Oracle*[4]). The English-Arabic data exhibits a slightly higher upper bound of 5.5% for *Oracle*[4]. The upper bounds for AER on English-Romanian data are even higher (up to 17.7%), which indicates that the input alignments are significantly worse than others. This may be one of the main contributing factors to the lower improvement of ACME on English-Romanian in comparison to the other two language pairs.

Alignments	**GIZA++**	**SAHMM**
Oracle[2] (en-ch)	9.4	8.4
Oracle[4] (en-ch)	4.7	
Oracle[2] (en-ar)	9.8	11.1
Oracle[4] (en-ar)	5.5	
Oracle[2] (en-ro)	15.4	17.7
Oracle[4] (en-ro)	11.3	

Table 7: Oracle Upper Bounds on AER for Alignment Combination

5.4 MT Evaluation

To determine the contribution of improved alignment in an external application, we examined the improvement in an off-the-shelf phrase-based MT system Pharaoh (Koehn, 2004) on both Chinese and Arabic data. In these experiments, all components of the MT system were kept the same except for the component that generates a phrase table from a given alignment.

The input alignments were generated using GIZA++ and SAHMM on 107K (44K) sentence pairs for Chinese (Arabic). ACME (with English POS partitioning) combines alignments using model parameters learned from the corresponding manually aligned data. MT output is evaluated using the standard MT evaluation metric BLEU (Papineni et al., 2002).[10] Table 8 presents the BLEU scores on

[9] If the input aligners agree on a particular link, that decision is taken as the final output in computing the upper bound.

[10] We used the NIST script (version 11a) with its default set-

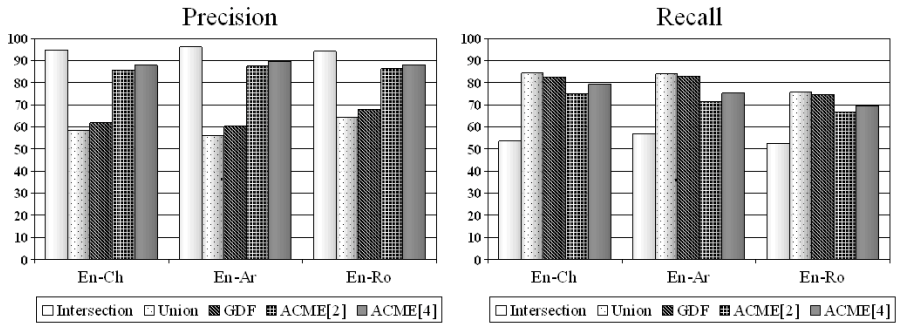


Figure 2: Precision and Recall Scores for GIZA++ and ACME Using 2 and 4 Input Alignments.

MTEval'03 data for 5 different Pharaoh runs, one for each alignment. The parameters of the MT system were optimized on MTEval'02 data using minimum error rate training (Och, 2003).

For the language model, the SRI Language Modeling Toolkit was used to train a trigram model with modified Kneser-Ney smoothing on 155M words of English newswire text, mostly from the Xinhua portion of the Gigaword corpus. During decoding, the number of English phrases per FL phrase was limited to 100 and the distortion of phrases was limited by 4. Based on the observations in (Koehn et al., 2003), we also limited the phrase length to 3 for computational reasons.

Alignment	Chinese	Arabic
GIZA++(union)	22.66	41.72
GIZA++(gdf)	23.79	43.82
GIZA++(int)	23.97	42.76
ACME[2]	25.20	44.94
ACME[4]	**25.59**	**45.54**

Table 8: Evaluation of Pharaoh with Different Initial Alignments using BLEU (in percentages)

For both languages, ACME[2] and ACME[4] outperform the other three alignment combination techniques. ACME[4], for instance, yields the BLEU scores of 25.59% for Chinese and 45.54% for Arabic—an absolute 1.6-1.7% BLEU point increase over the best of the other three alignment combinations. The differences between the BLEU scores for ACME and the other three BLEU scores are statistically significant, using a significance test with bootstrap resampling (Zhang et al., 2004).

6 Related Work

ME models have been previously applied to several NLP problems, including word alignments. For instance, the IBM models (Brown et al., 1993) can be improved by adding more context dependencies into the translation model using a ME framework rather than using only $p(f_j|e_i)$ (Garcia-Varea et al., 2002). In a later study, Och and Ney (2003) present a log-linear combination of the HMM and IBM Model 4 that produces better alignments than either of those. The major advantage of these two methods is that they do not require manually annotated data.

The alignment process can be modeled as a product of a transition model and an observation model, where ME models the observations (Ittycheriah and Roukos, 2005). Significant improvements are reported using this approach but the need for large manually aligned data is a bottleneck. An alternative ME approach models alignment directly as a log-linear combination of feature functions (Liu et al., 2005). Moore (2005) and Taskar et al. (2005) represent alignments with several feature functions that are then combined in a weighted sum to model word alignments. Once a confidence score is assigned to all links, a non-trivial search is invoked to find the best alignment using the scores associated with the links. The major difference between these approaches and that of ACME is that we use the ME model to predict the correct class for each alignment link independently using outputs of existing alignment systems, instead of generating them from scratch at the level of the whole sentence, thus eliminating the need for an exhaustive search over all possible alignments, i.e., previous approaches work globally while ACME is a localized model. A discussion of these two contrasting approaches can be found in (Tillmann and Zhang, 2005).

A recent attempt to combine outputs of different alignments views the combination problem as a classifier ensemble in the neural network framework

tings: case-insensitive matching of n-grams up to $n = 4$, and the shortest reference sentence for the brevity penalty.

(Ayan et al., 2005). However, this method is subject to the unpredictability of random network initialization, whereas ACME is guaranteed to find the model that maximizes the likelihood of training data.

7 Conclusions

We presented a new approach, ACME, to combining the outputs of different word alignment systems by reducing the combination problem to the level of alignment links and using a maximum entropy model to learn whether a particular alignment link is included in the final alignment.

Our results indicate that ACME yields significant relative error reduction over the input alignments and their heuristic-based combinations on three different language pairs. Moreover, ACME provides similar relative improvements for different sizes of training data for the input alignment systems. We have also shown that using a higher number of input alignments, and partitioning the training data into disjoint subsets and learning a different model for each partition yield further improvements.

We have tested impact of the reduced AER on MT and have shown that alignments generated by ACME yield statistically significant improvements in BLEU scores in two different languages, even if we don't employ a POS tagger on the FL side. However, additional studies are needed to investigate why huge improvements in AER result in relatively smaller improvements in BLEU scores.

Because ACME is a supervised learning approach, it requires annotated data; however, our experiments have shown that significant improvements can be obtained using a small set of annotated data.

Acknowledgments This work has been supported, in part, under ONR MURI Contract FCPO.810548265 and the GALE program of the Defense Advanced Research Projects Agency, Contracts No. HR0011-06-2-0001. We also thank anonymous reviewers for their helpful comments.

References

Necip F. Ayan, Bonnie J. Dorr, and Christof Monz. 2005. Neuralign: Combining word alignments using neural networks. In *Proceedings of EMNLP'2005*, pages 65–72.

Adam L. Berger, Stephan A. Della-Pietra, and Vincent J. Della-Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1).

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

J. N. Darroch and D. Ratcliff. 1972. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43:1470–1480.

Michael Fleischman, Namhee Kwon, and Eduard Hovy. 2003. Maximum entropy models for framenet classification. In *Proceedings of EMNLP'2003*.

Ismael Garcia-Varea, Franz Josef Och, Hermann Ney, and Francisco Casacuberta. 2002. Improving alignment quality in statistical machine translation using context-dependent maximum entropy models. In *Proceedings of COLING'2002*.

Abraham Ittycheriah and Salim Roukos. 2005. A maximum entropy word aligner for arabic-english machine translation. In *Proceedings of EMNLP'2005*.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL'2003*.

Philipp Koehn. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation. In *Proceedings of AMTA'2004*.

Yang Liu, Qun Liu, and Shouxun Lin. 2005. Log-linear models for word alignment. In *Proceedings of ACL'2005*.

Adam Lopez and Philip Resnik. 2005. Improved HMM alignment models for languages with scarce resources. In *Proceedings of the ACL'2005 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 83–86.

Rada Mihalcea and Ted Pedersen. 2003. An evaluation exercise for word alignment. In *Proceedings of the HLT-NAACL'2003 Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 1–10.

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

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. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):9–51, March.

Franz J. Och. 2000. GIZA++: Training of statistical translation models. Technical report, RWTH Aachen, University of Technology.

Franz J. Och. 2002. Yet another maxent toolkit: YASMET. Available at http://www.fjoch.com/YASMET.html.

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.

Adwait Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

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

Christoph Tillmann and Tong Zhang. 2005. A localized prediction model for statistical machine translation. In *Proceedings of ACL'2005*.

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 LREC'2004*, pages 2051–2054.

Alignment by Agreement

Percy Liang
UC Berkeley
Berkeley, CA 94720
pliang@cs.berkeley.edu

Ben Taskar
UC Berkeley
Berkeley, CA 94720
taskar@cs.berkeley.edu

Dan Klein
UC Berkeley
Berkeley, CA 94720
klein@cs.berkeley.edu

Abstract

We present an unsupervised approach to symmetric word alignment in which two simple asymmetric models are trained jointly to maximize a combination of data likelihood and agreement between the models. Compared to the standard practice of intersecting predictions of independently-trained models, joint training provides a 32% reduction in AER. Moreover, a simple and efficient pair of HMM aligners provides a 29% reduction in AER over symmetrized IBM model 4 predictions.

1 Introduction

Word alignment is an important component of a complete statistical machine translation pipeline (Koehn et al., 2003). The classic approaches to unsupervised word alignment are based on IBM models 1–5 (Brown et al., 1994) and the HMM model (Ney and Vogel, 1996) (see Och and Ney (2003) for a systematic comparison). One can classify these six models into two groups: sequence-based models (models 1, 2, and HMM) and fertility-based models (models 3, 4, and 5).[1] Whereas the sequence-based models are tractable and easily implemented, the more accurate fertility-based models are intractable and thus require approximation methods which are difficult to implement. As a result, many practitioners use the complex GIZA++ software package (Och and Ney, 2003) as a black box, selecting model 4 as a good compromise between alignment quality and efficiency.

Even though the fertility-based models are more accurate, there are several reasons to consider avenues for improvement based on the simpler and faster sequence-based models. First, even with the highly optimized implementations in GIZA++, models 3 and above are still very slow to train. Second, we seem to have hit a point of diminishing returns with extensions to the fertility-based models. For example, gains from the new model 6 of Och and Ney (2003) are modest. When models are too complex to reimplement, the barrier to improvement is raised even higher. Finally, the fertility-based models are asymmetric, and symmetrization is commonly employed to improve alignment quality by intersecting alignments induced in each translation direction. It is therefore natural to explore models which are designed from the start with symmetry in mind.

In this paper, we introduce a new method for word alignment that addresses the three issues above. Our development is motivated by the observation that intersecting the predictions of two directional models outperforms each model alone. Viewing intersection as a way of finding predictions that both models agree on, we take the agreement idea one step further. The central idea of our approach is to not only make the predictions of the models agree at test time, but also encourage agreement during training. We define an intuitive objective function which incor-

[1] IBM models 1 and 2 are considered sequence-based models because they are special cases of HMMs with transitions that do not depend on previous states.

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 104–111,
New York, June 2006.

porates both data likelihood and a measure of agreement between models. Then we derive an EM-like algorithm to maximize this objective function. Because the E-step is intractable in our case, we use a heuristic approximation which nonetheless works well in practice.

By jointly training two simple HMM models, we obtain 4.9% AER on the standard English-French Hansards task. To our knowledge, this is the lowest published unsupervised AER result, and it is competitive with supervised approaches. Furthermore, our approach is very practical: it is no harder to implement than a standard HMM model, and joint training is no slower than the standard training of two HMM models. Finally, we show that word alignments from our system can be used in a phrase-based translation system to modestly improve BLEU score.

2 Alignment models: IBM 1, 2 and HMM

We briefly review the sequence-based word alignment models (Brown et al., 1994; Och and Ney, 2003) and describe some of the choices in our implementation. All three models are generative models of the form $p(\mathbf{f} \mid \mathbf{e}) = \sum_{\mathbf{a}} p(\mathbf{a}, \mathbf{f} \mid \mathbf{e})$, where $\mathbf{e} = (e_1, \ldots, e_I)$ is the English sentence, $\mathbf{f} = (f_1, \ldots, f_J)$ is the French sentence, and $\mathbf{a} = (a_1, \ldots, a_J)$ is the (asymmetric) alignment which specifies the position of an English word aligned to each French word. All three models factor in the following way:

$$p(\mathbf{a}, \mathbf{f} \mid \mathbf{e}) = \prod_{j=1}^{J} p_{\mathrm{d}}(a_j \mid a_{j_-}, j) p_{\mathrm{t}}(f_j \mid e_{a_j}), \quad (1)$$

where j_- is the position of the last non-null-aligned French word before position j.[2]

The translation parameters $p_{\mathrm{t}}(f_j \mid e_{a_j})$ are parameterized by an (unsmoothed) lookup table that stores the appropriate local conditional probability distributions. The distortion parameters $p_{\mathrm{d}}(a_j = i' \mid a_{j_-} = i)$ depend on the particular model (we write $a_j = 0$ to denote the event that the j-th French word is null-aligned):

$$\begin{aligned} p_{\mathrm{d}}(a_j = 0 \mid a_{j_-} = i) &= p_0 \\ p_{\mathrm{d}}(a_j = i' \neq 0 \mid a_{j_-} = i) &\propto \\ (1 - p_0) \cdot &\begin{cases} 1 & \text{(IBM 1)} \\ c(i' - \lfloor \frac{jI}{J} \rfloor) & \text{(IBM 2)} \\ c(i' - i) & \text{(HMM),} \end{cases} \end{aligned}$$

where p_0 is the null-word probability and $c(\cdot)$ contains the distortion parameters for each offset argument. We set the null-word probability $p_0 = \frac{1}{I+1}$ depending on the length of the English sentence, which we found to be more effective than using a constant p_0.

In model 1, the distortion $p_{\mathrm{d}}(\cdot \mid \cdot)$ specifies a uniform distribution over English positions. In model 2, $p_{\mathrm{d}}(\cdot \mid \cdot)$ is still independent of a_{j_-}, but it can now depend on j and i' through $c(\cdot)$. In the HMM model, there is a dependence on $a_{j_-} = i$, but only through $c(i - i')$.

We parameterize the distortion $c(\cdot)$ using a multinomial distribution over 11 offset buckets $c(\leq -5), c(-4), \ldots, c(4), c(\geq 5)$.[3] We use three sets of distortion parameters, one for transitioning into the first state, one for transitioning out of the last state, and one for all other transitions. This works better than using a single set of parameters or ignoring the transitions at the two ends.

3 Training by agreement

To motivate our joint training approach, we first consider the standard practice of intersecting alignments. While the English and French sentences play a symmetric role in the word alignment task, sequence-based models are asymmetric: they are generative models of the form $p(\mathbf{f} \mid \mathbf{e})$ (E→F), or $p(\mathbf{e} \mid \mathbf{f})$ (F→E) by reversing the roles of source and target. In general, intersecting the alignment predictions of two independently-trained directional models reduces AER, e.g., from 11% to 7% for HMM models (Table 2). This suggests that two models make different types of errors that can be eliminated upon intersection. Figure 1 (top) shows a common type of error that intersection can partly remedy. In

[2] The dependence on a_{j_-} can in fact be implemented as a first-order HMM (see Och and Ney (2003)).

[3] For each sentence, the probability mass of each of the two end buckets $c(\leq -5)$ or $c(\geq 5)$ is uniformly divided among those valid offsets.

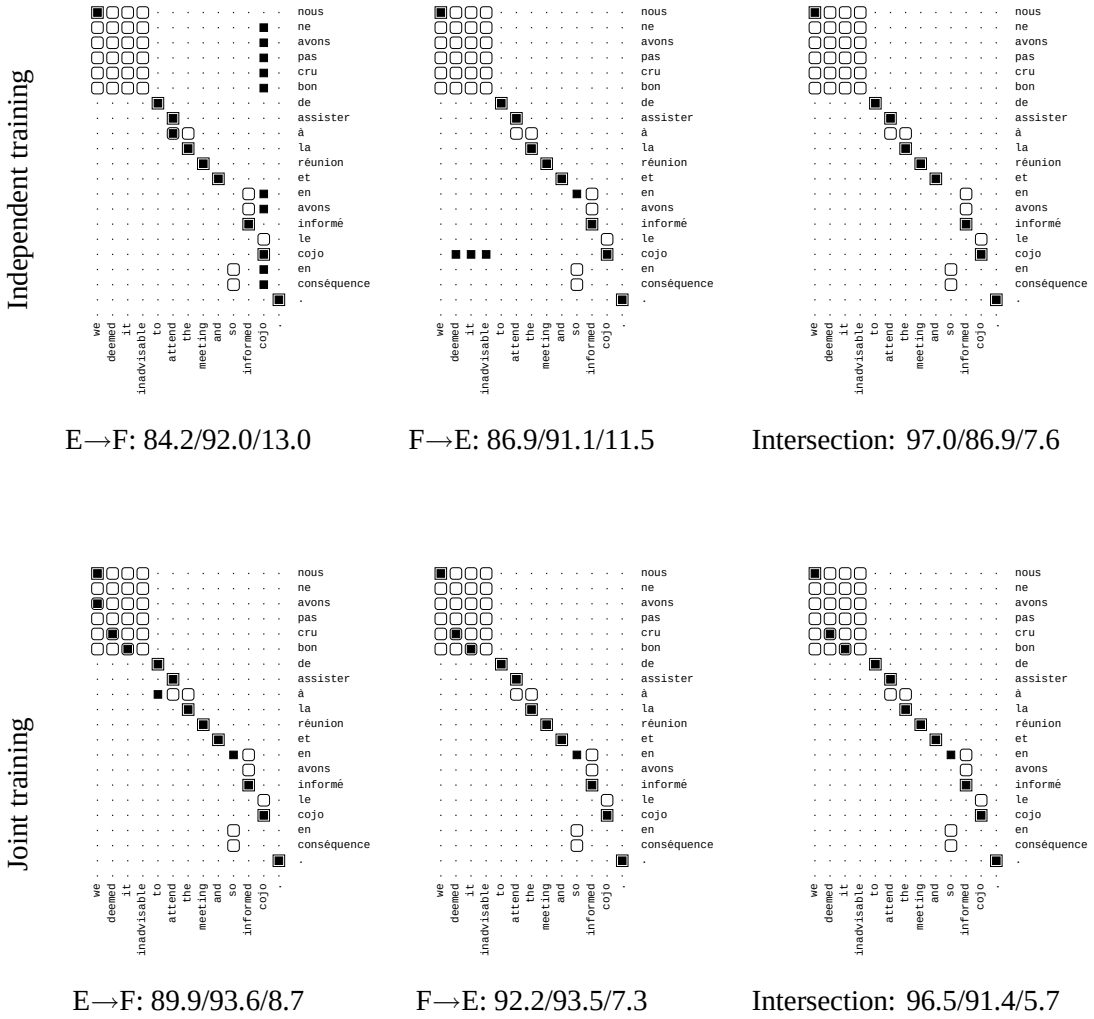


Figure 1: An example of the Viterbi output of a pair of independently trained HMMs (top) and a pair of jointly trained HMMs (bottom), both trained on 1.1 million sentences. Rounded boxes denote possible alignments, square boxes are sure alignments, and solid boxes are model predictions. For each model, the overall Precision/Recall/AER on the development set is given. See Section 4 for details.

this example, *COJO* is a rare word that becomes a garbage collector (Moore, 2004) for the models in both directions. Intersection eliminates the spurious alignments, but at the expense of recall.

Intersection after training produces alignments that both models agree on. The joint training procedure we describe below builds on this idea by encouraging the models to agree during training. Consider the output of the jointly trained HMMs in Figure 1 (bottom). The garbage-collecting rare word is no longer a problem. Not only are the individual E→F and F→E jointly-trained models better than their independently-trained counterparts, the jointly-trained intersected model also provides a significant overall gain over the independently-trained intersected model. We maintain both high precision and recall.

Before we introduce the objective function for joint training, we will write the two directional models in a symmetric way so that they share the same

alignment spaces. We first replace the asymmetric alignments $\mathbf{a}$ with a set of indicator variables for each potential alignment edge (i, j): $\mathbf{z} = \{z_{ij} \in \{0, 1\} : 1 \leq i \leq I, 1 \leq j \leq J\}$. Each $\mathbf{z}$ can be thought of as an element in the set of *generalized alignments*, where any subset of word pairs may be aligned (Och and Ney, 2003). Sequence-based models $p(\mathbf{a} \mid \mathbf{e}, \mathbf{f})$ induce a distribution over $p(\mathbf{z} \mid \mathbf{e}, \mathbf{f})$ by letting $p(\mathbf{z} \mid \mathbf{e}, \mathbf{f}) = 0$ for any $\mathbf{z}$ that does not correspond to any $\mathbf{a}$ (i.e., if $\mathbf{z}$ contains many-to-one alignments).

We also introduce the more compact notation $\mathbf{x} = (\mathbf{e}, \mathbf{f})$ to denote an input sentence pair. We put arbitrary distributions $p(\mathbf{e})$ and $p(\mathbf{f})$ to remove the conditioning, noting that this has no effect on the optimization problem in the next section. We can now think of the two directional sequence-based models as each inducing a distribution over the same space of sentence pairs and alignments $(\mathbf{x}, \mathbf{z})$:

$$p_1(\mathbf{x}, \mathbf{z}; \theta_1) = p(\mathbf{e})p(\mathbf{a}, \mathbf{f} \mid \mathbf{e}; \theta_1)$$
$$p_2(\mathbf{x}, \mathbf{z}; \theta_2) = p(\mathbf{f})p(\mathbf{a}, \mathbf{e} \mid \mathbf{f}; \theta_2).$$

3.1 A joint objective

In the next two sections, we describe how to jointly train the two models using an EM-like algorithm. We emphasize that this technique is quite general and can be applied in many different situations where we want to couple two tractable models over input $\mathbf{x}$ and output $\mathbf{z}$.

To train two models $p_1(\mathbf{x}, \mathbf{z}; \theta_1)$ and $p_2(\mathbf{x}, \mathbf{z}; \theta_2)$ independently, we maximize the data likelihood $\prod_{\mathbf{x}} p_k(\mathbf{x}; \theta_k) = \prod_{\mathbf{x}} \sum_{\mathbf{z}} p_k(\mathbf{x}, \mathbf{z}; \theta_k)$ of each model separately, $k \in \{1, 2\}$:

$$\max_{\theta_1, \theta_2} \sum_{\mathbf{x}} \left[\log p_1(\mathbf{x}; \theta_1) + \log p_2(\mathbf{x}; \theta_2)\right]. \quad (2)$$

Above, the summation over $\mathbf{x}$ enumerates the sentence pairs in the training data.

There are many possible ways to quantify agreement between two models. We chose a particularly simple and mathematically convenient measure — the probability that the alignments produced by the two models agree on an example $\mathbf{x}$:

$$\sum_{\mathbf{z}} p_1(\mathbf{z} \mid \mathbf{x}; \theta_1) p_2(\mathbf{z} \mid \mathbf{x}; \theta_2).$$

We add the (log) probability of agreement to the standard log-likelihood objective to couple the two models:

$$\max_{\theta_1, \theta_2} \sum_{\mathbf{x}} [\log p_1(\mathbf{x}; \theta_1) + \log p_2(\mathbf{x}; \theta_2) + \log \sum_{\mathbf{z}} p_1(\mathbf{z} \mid \mathbf{x}; \theta_1) p_2(\mathbf{z} \mid \mathbf{x}; \theta_2)]. \quad (3)$$

3.2 Optimization via EM

We first review the EM algorithm for optimizing a single model, which consists of iterating the following two steps:

$$\text{E}: \quad q(\mathbf{z}; \mathbf{x}) := p(\mathbf{z} \mid \mathbf{x}; \theta),$$
$$\text{M}: \quad \theta' := \operatorname*{argmax}_{\theta} \sum_{\mathbf{x}, \mathbf{z}} q(\mathbf{z}; \mathbf{x}) \log p(\mathbf{x}, \mathbf{z}; \theta).$$

In the E-step, we compute the posterior distribution of the alignments $q(\mathbf{z}; \mathbf{x})$ given the sentence pair $\mathbf{x}$ and current parameters θ. In the M-step, we use expected counts with respect to $q(\mathbf{z}; \mathbf{x})$ in the maximum likelihood update $\theta := \theta'$.

To optimize the objective in Equation 3, we can derive a similar and simple procedure. See the appendix for the derivation.

$$\text{E: } q(\mathbf{z}; \mathbf{x}) := \tfrac{1}{Z_{\mathbf{x}}} p_1(\mathbf{z} \mid \mathbf{x}; \theta_1) p_2(\mathbf{z} \mid \mathbf{x}; \theta_2),$$
$$\text{M: } \theta' = \operatorname*{argmax}_{\theta} \sum_{\mathbf{x}, \mathbf{z}} q(\mathbf{z}; \mathbf{x}) \log p_1(\mathbf{x}, \mathbf{z}; \theta_1) + \sum_{\mathbf{x}, \mathbf{z}} q(\mathbf{z}; \mathbf{x}) \log p_2(\mathbf{x}, \mathbf{z}; \theta_2),$$

where $Z_{\mathbf{x}}$ is a normalization constant. The M-step decouples neatly into two independent optimization problems, which lead to single model updates using the expected counts from $q(\mathbf{z}; \mathbf{x})$. To compute $Z_{\mathbf{x}}$ in the E-step, we must sum the product of two model posteriors over the set of possible $\mathbf{z}$s with nonzero probability under both models. In general, if both posterior distributions over the latent variables $\mathbf{z}$ decompose in the same tractable manner, as in the context-free grammar induction work of Klein and Manning (2004), the summation could be carried out efficiently, for example using dynamic programming. In our case, we would have to sum over the set of alignments where each word in English is aligned to at most one word in French and each word in French is aligned to at most one

word in English. Unfortunately, for even very simple models such as IBM 1 or 2, computing the normalization constant over this set of alignments is a $\#P$-complete problem, by a reduction from counting matchings in a bipartite graph (Valiant, 1979). We could perhaps attempt to compute q using a variety of approximate probabilistic inference techniques, for example, sampling or variational methods. With efficiency as our main concern, we opted instead for a simple heuristic procedure by letting q be a product of marginals:

$$q(\mathbf{z}; \mathbf{x}) := \prod_{i,j} p_1(z_{ij} \mid \mathbf{x}; \theta_1) p_2(z_{ij} \mid \mathbf{x}; \theta_2),$$

where each $p_k(z_{ij} \mid \mathbf{x}; \theta_k)$ is the posterior marginal probability of the (i, j) edge being present (or absent) in the alignment according to each model, which can be computed separately and efficiently.

Now the new E-step only requires simple marginal computations under each of the models. This procedure is very intuitive: edges on which the models disagree are discounted in the E-step because the product of the marginals $p_1(z_{ij} \mid \mathbf{x}; \theta_1) p_2(z_{ij} \mid \mathbf{x}; \theta_2)$ is small. Note that in general, this new procedure is not guaranteed to increase our joint objective. Nonetheless, our experimental results show that it provides an effective method of achieving model agreement and leads to significant accuracy gains over independent training.

3.3 Prediction

Once we have trained two models, either jointly or independently, we must decide how to combine those two models to predict alignments for new sentences.

First, let us step back to the case of one model. Typically, the Viterbi alignment $\operatorname{argmax}_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{x})$ is used. An alternative is to use posterior decoding, where we keep an edge (i, j) if the marginal edge posterior $p(z_{ij} \mid \mathbf{x})$ exceeds some threshold $0 < \delta < 1$. In symbols, $\mathbf{z} = \{z_{ij} = 1 : p(z_{ij} = 1 \mid \mathbf{x}) \geq \delta\}$.[4]

Posterior decoding has several attractive advantages over Viterbi decoding. Varying the threshold δ gives a natural way to tradeoff precision and recall. In fact, these posteriors could be used more directly in extracting phrases for phrase-based translation. Also, when we want to combine two models for prediction, finding the Viterbi alignment $\operatorname{argmax}_{\mathbf{z}} p_1(\mathbf{z} \mid \mathbf{x}) p_2(\mathbf{z} \mid \mathbf{x})$ is intractable for HMM models (by a reduction from quadratic assignment), and a hard intersection $\operatorname{argmax}_{\mathbf{z}_1} p_1(\mathbf{z}_1 \mid \mathbf{x}) \cap \operatorname{argmax}_{\mathbf{z}_2} p_2(\mathbf{z}_2 \mid \mathbf{x})$ might be too sparse. On the other hand, we can threshold the product of two edge posteriors quite easily: $\mathbf{z} = \{z_{ij} = 1 : p_1(z_{ij} = 1 \mid \mathbf{x}) p_2(z_{ij} = 1 \mid \mathbf{x}) \geq \delta\}$.

We noticed a 5.8% relative reduction in AER (for our best model) by using posterior decoding with a validation-set optimized threshold δ instead of using hard intersection of Viterbi alignments.

[4] See Matusov et al. (2004) for an alternative use of these marginals.

4 Experiments

We tested our approach on the English-French Hansards data from the NAACL 2003 Shared Task, which includes a training set of 1.1 million sentences, a validation set of 37 sentences, and a test set of 447 sentences. The validation and test sentences have been hand-aligned (see Och and Ney (2003)) and are marked with both *sure* and *possible* alignments. Using these alignments, *alignment error rate* (AER) is calculated as:

$$\left(1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}\right) \times 100\%,$$

where A is a set of proposed edges, S is the sure gold edges, and P is the possible gold edges.

As a preprocessing step, we lowercased all words. Then we used the validation set and the first 100 sentences of the test set as our development set to tune our models. Lastly, we ran our models on the last 347 sentences of the test set to get final AER results.

4.1 Basic results

We trained models 1, 2, and HMM on the Hansards data. Following past work, we initialized the translation probabilities of model 1 uniformly over word pairs that occur together in some sentence pair. Models 2 and HMM were initialized with uniform distortion probabilities and model 1 translation probabilities. Each model was trained for 5 iterations, using the same training regimen as in Och and Ney (2003).

Model	Indep.	Joint	Reduction
10K sentences			
Model 1	27.4	23.6	13.8
Model 2	18.2	14.9	18.5
HMM	12.1	8.4	30.6
100K sentences			
Model 1	21.5	19.2	10.9
Model 2	13.1	10.2	21.7
HMM	8.0	5.3	33.1
1.1M sentences			
Model 1	20.0	16.5	17.5
Model 2	11.4	9.2	18.8
HMM	6.6	5.2	21.5

Table 1: Comparison of AER between independent and joint training across different size training sets and different models, evaluated on the development set. The last column shows the relative reduction in AER.

Table 1 shows a summary of the performance of independently and jointly trained models under various training conditions. Quite remarkably, for all training data sizes and all of the models, we see an appreciable reduction in AER, especially on the HMM models. We speculate that since the HMM model provides a richer family of distributions over alignments than either models 1 or 2, we can learn to synchronize the predictions of the two models, whereas models 1 and 2 have a much more limited capacity to synchronize.

Table 2 shows the HMM models compared to model 4 alignments produced by GIZA++ on the test set. Our jointly trained model clearly outperforms not only the standard HMM but also the more complex IBM 4 model. For these results, the threshold used for posterior decoding was tuned on the development set. "GIZA HMM" and "HMM, indep" are the same algorithm but differ in implementation details. The E→F and F→E models benefit a great deal by moving from independent to joint training, and the combined models show a smaller improvement.

Our best performing model differs from standard IBM word alignment models in two ways. First and most importantly, we use joint training instead of independent training, which gives us a huge boost. The second change, which is more minor and orthogonal, is using posterior decoding instead of Viterbi decoding, which also helps performance for model 2 and HMM, but not model 1. Table 3 quantifies the contribution of each of these two dimensions.

Model	E→F	F→E	Combined
GIZA HMM	11.5	11.5	7.0
GIZA Model 4	8.9	9.7	6.9
HMM, indep	11.2	11.5	7.2
HMM, joint	6.1	6.6	**4.9**

Table 2: Comparison of test set AER between various models trained on the full 1.1 million sentences.

Model	I+V	I+P	J+V	J+P
10K sentences				
Model 1	29.4	27.4	**22.7**	23.6
Model 2	20.1	18.2	16.5	**14.9**
HMM	15.2	12.1	8.9	**8.4**
100K sentences				
Model 1	22.9	21.5	**18.6**	19.2
Model 2	15.1	13.1	12.9	**10.2**
HMM	9.2	8.0	6.0	**5.3**
1.1M sentences				
Model 1	20.0	19.4	**16.5**	17.3
Model 2	12.7	11.4	11.6	**9.2**
HMM	7.6	6.6	5.7	**5.2**

Table 3: Contributions of using joint training versus independent training and posterior decoding (with the optimal threshold) instead of Viterbi decoding, evaluated on the development set.

Posterior decoding In our results, we have tuned our threshold to minimize AER. It turns out that AER is relatively insensitive to the threshold as Figure 2 shows. There is a large range from 0.2 to 0.5 where posterior decoding outperforms Viterbi decoding.

Initialization and convergence In addition to improving performance, joint training also enjoys certain robustness properties. Specialized initialization is absolutely crucial for an independently-trained

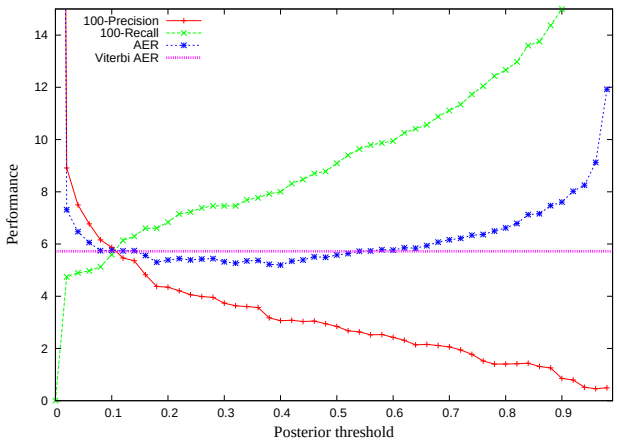


Figure 2: The precision, recall, and AER as the threshold is varied for posterior decoding in a jointly trained pair of HMMs.

HMM model. If we initialize the HMM model with uniform translation parameters, the HMM converges to a completely senseless local optimum with AER above 50%. Initializing the HMM with model 1 parameters alleviates this problem.

On the other hand, if we jointly train two HMMs starting from a uniform initialization, the HMMs converge to a surprisingly good solution. On the full training set, training two HMMs jointly from uniform initialization yields 5.7% AER, only slightly higher than 5.2% AER using model 1 initialization. We suspect that the agreement term of the objective forces the two HMMs to avoid many local optima that each one would have on its own, since these local optima correspond to posteriors over alignments that would be very unlikely to agree. We also observed that jointly trained HMMs converged very quickly—in 5 iterations—and did not exhibit overfitting with increased iterations.

Common errors The major source of remaining errors are recall errors that come from the shortcomings of the HMM model. The E→F model gives 0 probability to any many-to-one alignments and the F→E model gives 0 probability to any one-to-many alignments. By enforcing agreement, the two models are effectively restricted to one-to-one (or zero) alignments. Posterior decoding is in principle capable of proposing many-to-many alignments, but these alignments occur infrequently since the posteriors are generally sharply peaked around the Viterbi alignment. In some cases, however, we do get one-to-many alignments in both directions.

Another common type of errors are precision errors due to the models overly-aggressively preferring alignments that preserve monotonicity. Our HMM model only uses 11 distortion parameters, which means distortions are not sensitive to the lexical context of the sentences. For example, in one sentence, *le* is incorrectly aligned to *the* as a monotonic alignment following another pair of correctly aligned words, and then the monotonicity is broken immediately following *le–the*. Here, the model is insensitive to the fact that alignments following articles tend to be monotonic, but alignments preceding articles are less so.

Another phenomenon is the insertion of "stepping stone" alignments. Suppose two edges (i, j) and $(i{+}4, j{+}4)$ have a very high probability of being included in an alignment, but the words between them are not good translations of each other. If the intervening English words were null-aligned, we would have to pay a big distortion penalty for jumping 4 positions. On the other hand, if the edge $(i{+}2, j{+}2)$ were included, that penalty would be mitigated. The translation cost for forcing that edge is smaller than the distortion cost.

4.2 BLEU evaluation

To see whether our improvement in AER also improves BLEU score, we aligned 100K English-French sentences from the Europarl corpus and tested on 3000 sentences of length 5–15. Using GIZA++ model 4 alignments and Pharaoh (Koehn et al., 2003), we achieved a BLEU score of 0.3035. By using alignments from our jointly trained HMMs instead, we get a BLEU score of 0.3051. While this improvement is very modest, we are currently investigating alternative ways of interfacing with phrase table construction to make a larger impact on translation quality.

5 Related Work

Our approach is similar in spirit to co-training, where two classifiers, complementary by the virtue of having different views of the data, are trained jointly to encourage agreement (Blum and Mitchell, 1998; Collins and Singer, 1999). One key difference

in our work is that we rely exclusively on data likelihood to guide the two models in an unsupervised manner, rather than relying on an initial handful of labeled examples.

The idea of exploiting agreement between two latent variable models is not new; there has been substantial previous work on leveraging the strengths of two complementary models. Klein and Manning (2004) combine two complementary models for grammar induction, one that models constituency and one that models dependency, in a manner broadly similar to the current work. Aside from investigating a different domain, one novel aspect of this paper is that we present a formal objective and a training algorithm for combining two generic models.

6 Conclusion

We have described an efficient and fully unsupervised method of producing state-of-the-art word alignments. By training two simple sequence-based models to agree, we achieve substantial error reductions over standard models. Our jointly trained HMM models reduce AER by 29% over test-time intersected GIZA++ model 4 alignments and also increase our robustness to varying initialization regimens. While AER is only a weak indicator of final translation quality in many current translation systems, we hope that more accurate alignments can eventually lead to improvements in the end-to-end translation process.

Acknowledgments We thank the anonymous reviewers for their comments.

References

Avrim Blum and Tom Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-training. In *Proceedings of the COLT 1998*.

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.

Michael Collins and Yoram Singer. 1999. Unsupervised Models for Named Entity Classification. In *Proceedings of EMNLP 1999*.

Abraham Ittycheriah and Salim Roukos. 2005. A maximum entropy word aligner for arabic-english machine translation. In *Proceedings of HLT-EMNLP*.

Dan Klein and Christopher D. Manning. 2004. Corpus-Based Induction of Syntactic Structure: Models of Dependency and Constituency. In *Proceedings of ACL 2004*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of HLT-NAACL 2003*.

E. Matusov, Zens. R., and H. Ney. 2004. Symmetric word alignments for statistical machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics*, August.

Robert C. Moore. 2004. Improving IBM Word Alignment Model 1. In *Proceedings of ACL 2004*.

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

L. G. Valiant. 1979. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201.

Appendix: Derivation of agreement EM

To simplify notation, we drop the explicit reference to the parameters θ. Lower bound the objective in Equation 3 by introducing a distribution $q(\mathbf{z};\mathbf{x})$ and using the concavity of $\log$:

$$\sum_{\mathbf{x}} \log p_1(\mathbf{x})p_2(\mathbf{x}) \sum_{\mathbf{z}} p_1(\mathbf{z} \mid \mathbf{x})p_2(\mathbf{z} \mid \mathbf{x}) \quad (4)$$

$$\geq \sum_{\mathbf{x},\mathbf{z}} q(\mathbf{z};\mathbf{x}) \log \frac{p_1(\mathbf{x})p_2(\mathbf{x})p_1(\mathbf{z} \mid \mathbf{x})p_2(\mathbf{z} \mid \mathbf{x})}{q(\mathbf{z};\mathbf{x})} \quad (5)$$

$$= \sum_{\mathbf{x},\mathbf{z}} q(\mathbf{z};\mathbf{x}) \log \frac{p_1(\mathbf{z} \mid \mathbf{x})p_2(\mathbf{z} \mid \mathbf{x})}{q(\mathbf{z};\mathbf{x})} + C \quad (6)$$

$$= \sum_{\mathbf{x},\mathbf{z}} q(\mathbf{z};\mathbf{x}) \log p_1(\mathbf{x},\mathbf{z})p_2(\mathbf{x},\mathbf{z}) + D, \quad (7)$$

where C depends only on θ but not q and D depends only q but not θ. The E-step chooses q given a fixed θ to maximize the lower bound. Equation 6 is exactly $\sum_{\mathbf{x}} -\mathrm{KL}(q||p_1p_2) + C$, which is maximized by setting q proportional to p_1p_2. The M-step chooses θ given a fixed q. Equation 7 decomposes into two separate optimization problems.

Word Alignment via Quadratic Assignment

Simon Lacoste-Julien
UC Berkeley, Berkeley, CA 94720
slacoste@cs.berkeley.edu

Ben Taskar
UC Berkeley, Berkeley, CA 94720
taskar@cs.berkeley.edu

Dan Klein
UC Berkeley, Berkeley, CA 94720
klein@cs.berkeley.edu

Michael I. Jordan
UC Berkeley, Berkeley, CA 94720
jordan@cs.berkeley.edu

Abstract

Recently, discriminative word alignment methods have achieved state-of-the-art accuracies by extending the range of information sources that can be easily incorporated into aligners. The chief advantage of a discriminative framework is the ability to score alignments based on arbitrary features of the matching word tokens, including orthographic form, predictions of other models, lexical context and so on. However, the proposed bipartite matching model of Taskar et al. (2005), despite being tractable and effective, has two important limitations. First, it is limited by the restriction that words have fertility of at most one. More importantly, first order correlations between consecutive words cannot be directly captured by the model. In this work, we address these limitations by enriching the model form. We give estimation and inference algorithms for these enhancements. Our best model achieves a relative AER reduction of 25% over the basic matching formulation, outperforming intersected IBM Model 4 without using any overly compute-intensive features. By including predictions of other models as features, we achieve AER of 3.8 on the standard Hansards dataset.

1 Introduction

Word alignment is a key component of most end-to-end statistical machine translation systems. The standard approach to word alignment is to construct directional generative models (Brown et al., 1990; Och and Ney, 2003), which produce a sentence in one language given the sentence in another language. While these models require sentence-aligned bitexts, they can be trained with no further supervision, using EM. Generative alignment models do, however, have serious drawbacks. First, they require extensive tuning and processing of large amounts of data which, for the better-performing models, is a non-trivial resource requirement. Second, conditioning on arbitrary features of the input is difficult; for example, we would like to condition on the orthographic similarity of a word pair (for detecting cognates), the presence of that pair in various dictionaries, the similarity of the frequency of its two words, choices made by other alignment systems, and so on.

Recently, Moore (2005) proposed a discriminative model in which pairs of sentences (e, f) and proposed alignments a are scored using a linear combination of arbitrary features computed from the tuples (a, e, f). While there are no restrictions on the form of the model features, the problem of finding the highest scoring alignment is very difficult and involves heuristic search. Moreover, the parameters of the model must be estimated using averaged perceptron training (Collins, 2002), which can be unstable. In contrast, Taskar et al. (2005) cast word alignment as a maximum weighted matching problem, in which each pair of words (e_j, f_k) in a sentence pair (e, f) is associated with a score $s_{jk}(e, f)$ reflecting the desirability of the alignment of that pair. Importantly, this problem is computationally tractable. The alignment for the sentence pair is the highest scoring matching under constraints (such as the constraint that matchings be one-to-one). The scoring model $s_{jk}(e, f)$ can be based on a rich feature set defined on word pairs (e_j, f_k) and their context, including measures of association, orthography, relative position, predictions of generative models, etc. The parameters of the model are estimated within the framework of large-margin estimation; in particular, the problem turns out to reduce to the

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 112–119,
New York, June 2006.

solution of a (relatively) small quadratic program (QP). The authors show that large-margin estimation is both more stable and more accurate than perceptron training.

While the bipartite matching approach is a useful first step in the direction of discriminative word alignment, for discriminative approaches to compete with and eventually surpass the most sophisticated generative models, it is necessary to consider more realistic underlying statistical models. Note in particular two substantial limitations of the bipartite matching model of Taskar et al. (2005): words have fertility of at most one, and there is no way to incorporate pairwise interactions among alignment decisions. Moving beyond these limitations—while retaining computational tractability—is the next major challenge for discriminative word alignment.

In this paper, we show how to overcome both limitations. First, we introduce a parameterized model that penalizes different levels of fertility. While this extension adds very useful expressive power to the model, it turns out not to increase the computational complexity of the aligner, for either the prediction or the parameter estimation problem. Second, we introduce a more thoroughgoing extension which incorporates first-order interactions between alignments of consecutive words into the model. We do this by formulating the alignment problem as a quadratic assignment problem (QAP), where in addition to scoring individual edges, we also define scores of pairs of edges that connect consecutive words in an alignment. The predicted alignment is the highest scoring quadratic assignment.

QAP is an NP-hard problem, but in the range of problem sizes that we need to tackle the problem can be solved efficiently. In particular, using standard off-the-shelf integer program solvers, we are able to solve the QAP problems in our experiments in under a second. Moreover, the parameter estimation problem can also be solved efficiently by making use of a linear relaxation of QAP for the min-max formulation of large-margin estimation (Taskar, 2004).

We show that these two extensions yield significant improvements in error rates when compared to the bipartite matching model. The addition of a fertility model improves the AER by 0.4. Modeling first-order interactions improves the AER by 1.8. Combining the two extensions results in an improvement in AER of 2.3, yielding alignments of better quality than intersected IBM Model 4. Moreover, including predictions of bi-directional IBM Model 4 and model of Liang et al. (2006) as features, we achieve an absolute AER of 3.8 on the English-French Hansards alignment task—the best AER result published on this task to date.

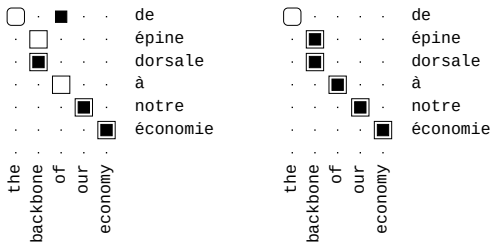
2 Models

We begin with a quick summary of the maximum weight bipartite matching model in (Taskar et al., 2005). More precisely, nodes $\mathcal{V} = \mathcal{V}^s \cup \mathcal{V}^t$ correspond to words in the "source" ($\mathcal{V}^s$) and "target" ($\mathcal{V}^t$) sentences, and edges $\mathcal{E} = \{jk : j \in \mathcal{V}^s, k \in \mathcal{V}^t\}$ correspond to alignments between word pairs.[1] The edge weights s_{jk} represent the degree to which word j in one sentence can be translated using the word k in the other sentence. The predicted alignment is chosen by maximizing the sum of edge scores. A matching is represented using a set of binary variables y_{jk} that are set to 1 if word j is assigned to word k in the other sentence, and 0 otherwise. The score of an assignment is the sum of edge scores: $s(\mathbf{y}) = \sum_{jk} s_{jk} y_{jk}$. For simplicity, let us begin by assuming that each word aligns to one or zero words in the other sentence; we revisit the issue of fertility in the next section. The maximum weight bipartite matching problem, $\arg\max_{\mathbf{y} \in \mathcal{Y}} s(\mathbf{y})$, can be solved using combinatorial algorithms for min-cost max-flow, expressed in a linear programming (LP) formulation as follows:

$$\begin{aligned} \max_{0 \le \mathbf{z} \le 1} \quad & \sum_{jk \in \mathcal{E}} s_{jk} z_{jk} \\ \text{s.t.} \quad & \sum_{j \in \mathcal{V}^s} z_{jk} \le 1, \ \forall k \in \mathcal{V}^t; \\ & \sum_{k \in \mathcal{V}^t} z_{jk} \le 1, \ \forall j \in \mathcal{V}^s, \end{aligned} \tag{1}$$

where the continuous variables z_{jk} are a relaxation of the corresponding binary-valued variables y_{jk}. This LP is guaranteed to have integral (and hence optimal) solutions for any scoring function $s(\mathbf{y})$ (Schrijver, 2003). Note that although the above LP can be used to compute alignments, combinatorial algorithms are generally more efficient. For

[1] The source/target designation is arbitrary, as the models considered below are all symmetric.



(a) (b)

Figure 2: An example fragment that requires fertility greater than one to correctly label. (a) The guess of the baseline M model. (b) The guess of the M+F fertility-augmented model.

example, in Figure 1(a), we show a standard construction for an equivalent min-cost flow problem. However, we build on this LP to develop our extensions to this model below. Representing the prediction problem as an LP or an integer LP provides a precise (and concise) way of specifying the model and allows us to use the large-margin framework of Taskar (2004) for parameter estimation described in Section 3.

For a sentence pair $\mathbf{x}$, we denote position pairs by $\mathbf{x}_{jk}$ and their scores as s_{jk}. We let $s_{jk} = \mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk})$ for some user provided feature mapping $\mathbf{f}$ and abbreviate $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{jk} y_{jk} \mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk})$. We can include in the feature vector the identity of the two words, their relative positions in their respective sentences, their part-of-speech tags, their string similarity (for detecting cognates), and so on.

2.1 Fertility

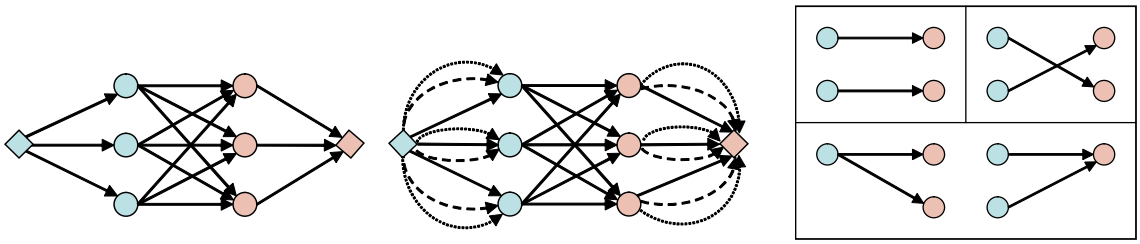
An important limitation of the model in Eq. (1) is that in each sentence, a word can align to at most one word in the translation. Although it is common that words have gold fertility zero or one, it is certainly not always true. Consider, for example, the bitext fragment shown in Figure 2(a), where *backbone* is aligned to the phrase *épine dorsal*. In this figure, outlines are gold alignments, square for sure alignments, round for possibles, and filled squares are target alignments (for details on gold alignments, see Section 4). When considering only the sure alignments on the standard Hansards dataset, 7 percent of the word occurrences have fertility 2, and 1 percent have fertility 3 and above; when considering the possible alignments high fertility is much more common—31 percent of the words have fertility 3 and above.

One simple fix to the original matching model is to increase the right hand sides for the constraints in Eq. (1) from 1 to D, where D is the maximum allowed fertility. However, this change results in an undesirable bimodal behavior, where maximum weight solutions either have all words with fertility 0 or D, depending on whether most scores $\mathbf{s}_{jk}$ are positive or negative. For example, if scores tend to be positive, most words will want to collect as many alignments as they are permitted. What the model is missing is a means for encouraging the common case of low fertility (0 or 1), while allowing higher fertility when it is licensed. This end can be achieved by introducing a penalty for having higher fertility, with the goal of allowing that penalty to vary based on features of the word in question (such as its frequency or identity).

In order to model such a penalty, we introduce indicator variables $z_{dj\bullet}$ (and $z_{d\bullet k}$) with the intended meaning: node j has fertility of at least d (and node k has fertility of at least d). In the following LP, we introduce a penalty of $\sum_{2 \le d \le D} s_{dj\bullet} z_{dj\bullet}$ for fertility of node j, where each term $s_{dj\bullet} \ge 0$ is the penalty increment for increasing the fertility from $d-1$ to d:

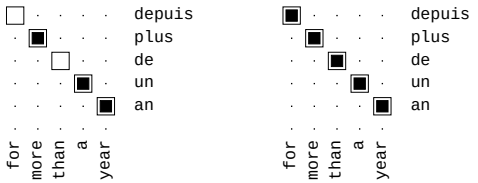
$$\max_{0 \le \mathbf{z} \le 1} \sum_{jk \in \mathcal{E}} s_{jk} z_{jk} \qquad (2)$$
$$- \sum_{j \in \mathcal{V}^s, 2 \le d \le D} s_{dj\bullet} z_{dj\bullet} - \sum_{k \in \mathcal{V}^t, 2 \le d \le D} s_{d\bullet k} z_{d\bullet k}$$
$$\text{s.t.} \quad \sum_{j \in \mathcal{V}^s} z_{jk} \le 1 + \sum_{2 \le d \le D} z_{d\bullet k}, \quad \forall k \in \mathcal{V}^t;$$
$$\sum_{k \in \mathcal{V}^t} z_{jk} \le 1 + \sum_{2 \le d \le D} z_{dj\bullet}, \quad \forall j \in \mathcal{V}^s.$$

We can show that this LP always has integral solutions by a reduction to a min-cost flow problem. The construction is shown in Figure 1(b). To ensure that the new variables have the intended semantics, we need to make sure that $s_{dj\bullet} \le s_{d'j\bullet}$ if $d \le d'$, so that the lower cost $z_{dj\bullet}$ is used before the higher cost $z_{d'j\bullet}$ to increase fertility. This restriction im-



(a) (b) (c)

Figure 1: (a) Maximum weight bipartite matching as min-cost flow. Diamond-shaped nodes represent flow source and sink. All edge capacities are 1, with edges between round nodes (j, k) have cost $-s_{jk}$, edges from source and to sink have cost 0. (b) Expanded min-cost flow graph with new edges from source and to sink that allow fertility of up to 3. The capacities of the new edges are 1 and the costs are 0 for solid edges from source and to sink, $s_{2j\bullet}$, $s_{2\bullet k}$ for dashed edges, and $s_{3j\bullet}$, $s_{3\bullet k}$ for dotted edges. (c) Three types of pairs of edges included in the QAP model, where the nodes on both sides correspond to consecutive words.



(a) (b)

Figure 3: An example fragment with a monotonic gold alignment. (a) The guess of the baseline M model. (b) The guess of the M+Q quadratic model.

plies that the penalty must be monotonic and convex as a function of the fertility.

To anticipate the results that we report in Section 4, adding fertility to the basic matching model makes the target alignment of the *backbone* example feasible and, in this case, the model correctly labels this fragment as shown in Figure 2(b).

2.2 First-order interactions

An even more significant limitation of the model in Eq. (1) is that the edges interact only indirectly through the competition induced by the constraints. Generative alignment models like the HMM model (Vogel et al., 1996) and IBM models 4 and above (Brown et al., 1990; Och and Ney, 2003) directly model correlations between alignments of consecutive words (at least on one side). For example, Figure 3 shows a bitext fragment whose gold alignment is strictly monotonic. This monotonicity is quite common – 46% of the words in the hand-aligned data diagonally follow a previous alignment in this way. We can model the common local alignment configurations by adding bonuses for pairs of edges. For example, strictly monotonic alignments can be encouraged by boosting the scores of edges of the form $\langle (j, k), (j + 1, k + 1) \rangle$. Another trend, common in English-French translation (7% on the hand-aligned data), is the local inversion of nouns and adjectives, which typically involves a pair of edges $\langle (j, k + 1), (j + 1, k) \rangle$. Finally, a word in one language is often translated as a phrase (consecutive sequence of words) in the other language. This pattern involves pairs of edges with the same origin on one side: $\langle (j, k), (j, k+1) \rangle$ or $\langle (j, k), (j+1, k) \rangle$. All three of these edge pair patterns are shown in Figure 1(c). Note that the set of such edge pairs $\mathcal{Q} = \{jklm : |j - l| \leq 1, |k - m| \leq 1\}$ is of linear size in the number of edges.

Formally, we add to the model variables z_{jklm} which indicate whether both edge jk and lm are in the alignment. We also add a corresponding score s_{jklm}, which we assume to be non-negative, since the correlations we described are positive. (Negative scores can also be used, but the resulting formulation we present below would be slightly different.) To enforce the semantics $z_{jklm} = z_{jk} z_{lm}$, we use a pair of constraints $z_{jklm} \leq z_{jk}; z_{jklm} \leq z_{lm}$. Since s_{jklm} is positive, at the optimum, $z_{jklm} =$

$\min(z_{jk}, z_{lm})$. If in addition z_{jk}, z_{lm} are integral (0 or 1), then $z_{jklm} = z_{jk} z_{lm}$. Hence, solving the following LP as an integer linear program will find the optimal quadratic assignment for our model:

$$\begin{aligned} \max_{0 \le \mathbf{z} \le 1} \quad & \sum_{jk \in \mathcal{E}} s_{jk} z_{jk} + \sum_{jklm \in \mathcal{Q}} s_{jklm} z_{jklm} \qquad (3) \\ \text{s.t.} \quad & \sum_{j \in \mathcal{V}^s} z_{jk} \le 1, \ \forall k \in \mathcal{V}^t; \\ & \sum_{k \in \mathcal{V}^t} z_{jk} \le 1, \ \forall j \in \mathcal{V}^s; \\ & z_{jklm} \le z_{jk}, \quad z_{jklm} \le z_{lm}, \ \forall jklm \in \mathcal{Q}. \end{aligned}$$

Note that we can also combine this extension with the fertility extension described above.

To once again anticipate the results presented in Section 4, the baseline model of Taskar et al. (2005) makes the prediction given in Figure 3(a) because the two missing alignments are atypical translations of common words. With the addition of edge pair features, the overall monotonicity pushes the alignment to that of Figure 3(b).

3 Parameter estimation

To estimate the parameters of our model, we follow the large-margin formulation of Taskar (2004). Our input is a set of training instances $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$, where each instance consists of a sentence pair $\mathbf{x}_i$ and a target alignment $\mathbf{y}_i$. We would like to find parameters $\mathbf{w}$ that predict correct alignments on the training data: $\mathbf{y}_i = \arg\max_{\bar{\mathbf{y}}_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}(\mathbf{x}_i, \bar{\mathbf{y}}_i)$ for each i, where $\mathcal{Y}_i$ is the space of matchings for the sentence pair $\mathbf{x}_i$.

In standard classification problems, we typically measure the error of prediction, $\ell(\mathbf{y}_i, \bar{\mathbf{y}}_i)$, using the simple 0-1 loss. In structured problems, where we are jointly predicting multiple variables, the loss is often more complex. While the F-measure is a natural loss function for this task, we instead chose a sensible surrogate that fits better in our framework: weighted Hamming distance, which counts the number of variables in which a candidate solution $\bar{\mathbf{y}}$ differs from the target output $\mathbf{y}$, with different penalty for false positives (c^+) and false negatives (c^-):

$$\ell(\mathbf{y}, \bar{\mathbf{y}}) = \sum_{jk} \left[c^+ (1 - y_{jk}) \bar{y}_{jk} + c^- (1 - \bar{y}_{jk}) y_{jk} \right].$$

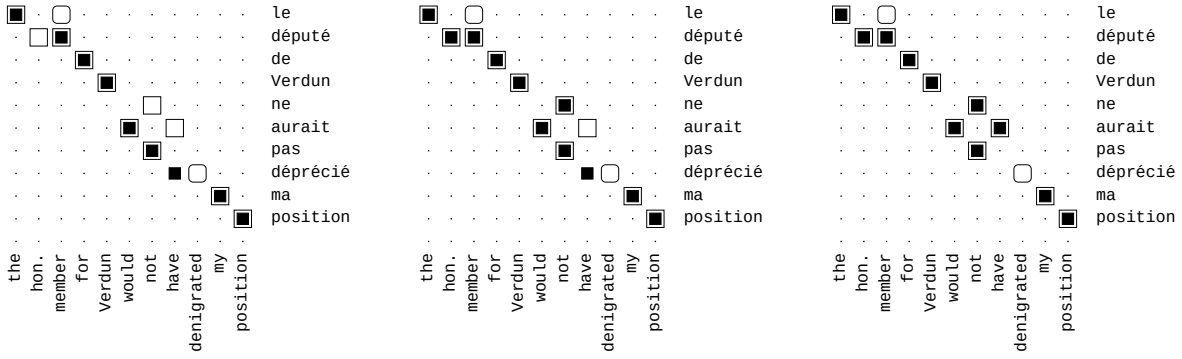
We use an SVM-like hinge upper bound on the loss $\ell(\mathbf{y}_i, \bar{\mathbf{y}}_i)$, given by $\max_{\bar{\mathbf{y}}_i \in \mathcal{Y}_i} [\mathbf{w}^\top \mathbf{f}_i(\bar{\mathbf{y}}_i) + \ell_i(\bar{\mathbf{y}}_i) - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i)]$, where $\ell_i(\bar{\mathbf{y}}_i) = \ell(\mathbf{y}_i, \bar{\mathbf{y}}_i)$, and $\mathbf{f}_i(\bar{\mathbf{y}}_i) = \mathbf{f}(\mathbf{x}_i, \bar{\mathbf{y}}_i)$. Minimizing this upper bound encourages the true alignment $\mathbf{y}_i$ to be optimal with respect to $\mathbf{w}$ for each instance i:

$$\min_{||\mathbf{w}|| \le \gamma} \sum_i \max_{\bar{\mathbf{y}}_i \in \mathcal{Y}_i} [\mathbf{w}^\top \mathbf{f}_i(\bar{\mathbf{y}}_i) + \ell_i(\bar{\mathbf{y}}_i)] - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i),$$

where γ is a regularization parameter.

In this form, the estimation problem is a mixture of continuous optimization over $\mathbf{w}$ and combinatorial optimization over $\mathbf{y}_i$. In order to transform it into a more standard optimization problem, we need a way to efficiently handle the *loss-augmented inference*, $\max_{\bar{\mathbf{y}}_i \in \mathcal{Y}_i} [\mathbf{w}^\top \mathbf{f}_i(\bar{\mathbf{y}}_i) + \ell_i(\bar{\mathbf{y}}_i)]$. This optimization problem has precisely the same form as the prediction problem whose parameters we are trying to learn — $\max_{\bar{\mathbf{y}}_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\bar{\mathbf{y}}_i)$ — but with an additional term corresponding to the loss function. Our assumption that the loss function decomposes over the edges is crucial to solving this problem. We omit the details here, but note that we can incorporate the loss function into the LPs for various models we described above and "plug" them into the large-margin formulation by converting the estimation problem into a quadratic problem (QP) (Taskar, 2004). This QP can be solved using any off-the-shelf solvers, such as MOSEK or CPLEX.[2] An important difference that comes into play for the estimation of the quadratic assignment models in Equation (3) is that inference involves solving an integer linear program, not just an LP. In fact the LP is a relaxation of the integer LP and provides an upper bound on the value of the highest scoring assignment. Using the LP relaxation for the large-margin QP formulation is an approximation, but as our experiments indicate, this approximation is very effective. At testing time, we use the integer LP to predict alignments. We have also experimented with using just the LP relaxation at testing time and then independently rounding each fractional edge value, which actually incurs no loss in alignment accuracy, as we discuss below.

[2] When training on 200 sentences, the QP we obtain contains roughly 700K variables and 300K constraints and is solved in roughly 10 minutes on a 2.8 GHz Pentium 4 machine. Aligning the whole training set with the fbw formulation takes a few seconds, whereas using the integer programming (for the QAP formulation) takes 1-2 minutes.



(a) (b) (c)

Figure 4: An example fragment with several multiple fertility sure alignments. (a) The guess of the M+Q model with maximum fertility of one. (b) The guess of the M+Q+F quadratic model with fertility two permitted. (c) The guess of the M+Q+F model with lexical fertility features.

4 Experiments

We applied our algorithms to word-level alignment using the English-French Hansards data from the 2003 NAACL shared task (Mihalcea and Pedersen, 2003). This corpus consists of 1.1M automatically aligned sentences, and comes with a validation set of 37 sentence pairs and a test set of 447 sentences. The validation and test sentences have been hand-aligned (see Och and Ney (2003)) and are marked with both *sure* and *possible* alignments. Using these alignments, *alignment error rate* (AER) is calculated as:

$$\left(1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}\right) \times 100\%.$$

Here, A is a set of proposed index pairs, S is the sure gold pairs, and P is the possible gold pairs. For example, in Figure 4, proposed alignments are shown against gold alignments, with open squares for sure alignments, rounded open squares for possible alignments, and filled black squares for proposed alignments.

The input to our algorithm is a small number of labeled examples. In order to make our results more comparable with Moore (2005), we split the original set into 200 training examples and 247 test examples. We also trained on only the first 100 to make our results more comparable with the experiments of Och and Ney (2003), in which IBM model 4 was tuned using 100 sentences. In all our experiments, we used a structured loss function that penalized false negatives 10 times more than false positives, where the value of 10 was picked by using a validation set. The regularization parameter γ was also chosen using the validation set.

4.1 Features and results

We parameterized all scoring functions s_{jk}, $s_{dj\bullet}$, $s_{d\bullet k}$ and s_{jklm} as weighted linear combinations of feature sets. The features were computed from the large unlabeled corpus of 1.1M automatically aligned sentences.

In the remainder of this section we describe the improvements to the model performance as various features are added. One of the most useful features for the basic matching model is, of course, the set of predictions of IBM model 4. However, computing these features is very expensive and we would like to build a competitive model that doesn't require them. Instead, we made significant use of IBM model 2 as a source of features. This model, although not very accurate as a predictive model, is simple and cheap to construct and it is a useful source of features.

The Basic Matching Model: Edge Features In the basic matching model of Taskar et al. (2005), called M here, one can only specify features on pairs of word tokens, i.e. alignment edges. These features

include word association, orthography, proximity, etc., and are documented in Taskar et al. (2005). We also augment those features with the predictions of IBM Model 2 run on the training and test sentences. We provided features for model 2 trained in each direction, as well as the intersected predictions, on each edge. By including the IBM Model 2 features, the performance of the model described in Taskar et al. (2005) on our test set (trained on 200 sentences) improves from 10.0 AER to 8.2 AER, outperforming unsymmetrized IBM Model 4 (but not intersected model 4).

As an example of the kinds of errors the baseline M system makes, see Figure 2 (where multiple fertility cannot be predicted), Figure 3 (where a preference for monotonicity cannot be modeled), and Figure 4 (which shows several multi-fertile cases).

The Fertility Model: Node Features To address errors like those shown in Figure 2, we increased the maximum fertility to two using the parameterized fertility model of Section 2.1. The model learns costs on the second flow arc for each word via features not of edges but of single words. The score of taking a second match for a word w was based on the following features: a bias feature, the proportion of times w's type was aligned to two or more words by IBM model 2, and the bucketed frequency of the word type. This model was called M+F. We also included a lexicalized feature for words which were common in our training set: whether w was ever seen in a multiple fertility alignment (more on this feature later). This enabled the system to learn that certain words, such as the English *not* and French verbs like *aurait* commonly participate in multiple fertility configurations.

Figure 5 show the results using the fertility extension. Adding fertility lowered AER from 8.5 to 8.1, though fertility was even more effective in conjunction with the quadratic features below. The M+F setting was even able to correctly learn some multiple fertility instances which were not seen in the training data, such as those shown in Figure 2.

The First-Order Model: Quadratic Features With or without the fertility model, the model makes mistakes such as those shown in Figure 3, where atypical translations of common words are not chosen despite their local support from adjacent edges. In the quadratic model, we can associate features with pairs of edges. We began with features which identify each specific pattern, enabling trends of monotonicity (or inversion) to be captured. We also added to each edge pair the fraction of times that pair's pattern (monotonic, inverted, one to two) occurred according each version of IBM model 2 (forward, backward, intersected).

Figure 5 shows the results of adding the quadratic model. M+Q reduces error over M from 8.5 to 6.7 (and fixes the errors shown in Figure 3). When both the fertility and quadratic extensions were added, AER dropped further, to 6.2. This final model is even able to capture the diamond pattern in Figure 4; the adjacent cycle of alignments is reinforced by the quadratic features which boost adjacency. The example in Figure 4 shows another interesting phenomenon: the multi-fertile alignments for *not* and *député* are learned even without lexical fertility features (Figure 4b), because the Dice coefficients of those words with their two alignees are both high. However the surface association of *aurait* with *have* is much higher than with *would*. If, however, lexical features are added, *would* is correctly aligned as well (Figure 4c), since it is observed in similar periphrastic constructions in the training set.

We have avoided using expensive-to-compute features like IBM model 4 predictions up to this point. However, if these are available, our model can improve further. By adding model 4 predictions to the edge features, we get a relative AER reduction of 27%, from 6.5 to 4.5. By also including as features the posteriors of the model of Liang et al. (2006), we achieve AER of 3.8, and 96.7/95.5 precision/recall.

It is comforting to note that in practice, the burden of running an integer linear program at test time can be avoided. We experimented with using just the LP relaxation and found that on the test set, only about 20% of sentences have fractional solutions and only 0.2% of all edges are fractional. Simple rounding[3] of each edge value in the LP solution achieves the same AER as the integer LP solution, while using about a third of the computation time on average.

[3] We slightly bias the system on the recall side by rounding 0.5 up, but this doesn't yield a noticeable difference in the results.

Model	Prec	Rec	AER
Generative			
IBM 2 (E→F)	73.6	87.7	21.7
IBM 2 (F→E)	75.4	87.0	20.6
IBM 2 (intersected)	90.1	80.4	14.3
IBM 4 (E→F)	90.3	92.1	9.0
IBM 4 (F→E)	90.8	91.3	9.0
IBM 4 (intersected)	98.0	88.1	6.5
Discriminative (100 sentences)			
Matching (M)	94.1	88.5	8.5
M + Fertility (F)	93.9	89.4	8.1
M + Quadratic (Q)	94.4	91.9	6.7
M + F + Q	94.8	92.5	6.2
M + F + Q + IBM4	96.4	94.4	4.5
Discriminative (200 sentences)			
Matching (M)	93.4	89.7	8.2
M + Fertility (F)	93.6	90.1	8.0
M + Quadratic (Q)	95.0	91.1	6.8
M + F + Q	95.2	92.4	6.1
M + F + Q + IBM4	96.0	95.0	4.4

Figure 5: AER on the Hansards task.

5 Conclusion

We have shown that the discriminative approach to word alignment can be extended to allow flexible fertility modeling and to capture first-order interactions between alignments of consecutive words. These extensions significantly enhance the expressive power of the discriminative approach; in particular, they make it possible to capture phenomena of monotonicity, local inversion and contiguous fertility trends—phenomena that are highly informative for alignment. They do so while remaining computationally efficient in practice both for prediction and for parameter estimation.

Our best model achieves a relative AER reduction of 25% over the basic matching formulation, beating intersected IBM Model 4 without the use of any compute-intensive features. Including Model 4 predictions as features, we achieve a further relative AER reduction of 32% over intersected Model 4 alignments. By also including predictions of another model, we drive AER down to 3.8. We are currently investigating whether the improvement in AER results in better translation BLEU score. Allowing higher fertility and optimizing a recall biased cost function provide a significant increase in recall relative to the intersected IBM model 4 (from 88.1% to 94.4%), with only a small degradation in precision. We view this as a particularly promising aspect of our work, given that phrase-based systems such as Pharaoh (Koehn et al., 2003) perform better with higher recall alignments.

References

P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

M. Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003*.

P. Liang, B. Taskar, and D. Klein. 2006. Alignment by agreement. In *HLT-NAACL*.

R. Mihalcea and T. Pedersen. 2003. An evaluation exercise for word alignment. In *Proceedings of the HLT-NAACL 2003 Workshop, Building and Using parallel Texts: Data Driven Machine Translation and Beyond*, pages 1–6, Edmonton, Alberta, Canada.

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

F. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–52.

A. Schrijver. 2003. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer.

B. Taskar, S. Lacoste-Julien, and D. Klein. 2005. A discriminative matching approach to word alignment. In *EMNLP*.

B. Taskar. 2004. *Learning Structured Prediction Models: A Large Margin Approach*. Ph.D. thesis, Stanford University.

S. Vogel, H. Ney, and C. Tillmann. 1996. HMM-based word alignment in statistical translation. In *COLING 16*, pages 836–841.

An Empirical Study of the Behavior of Active Learning for Word Sense Disambiguation

[1] Jinying Chen, [1] Andrew Schein, [1] Lyle Ungar, [2] Martha Palmer
[1] Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA, 19104
{jinying,ais,ungar}@cis.upenn.edu
[2] Linguistic Department
University of Colorado
Boulder, CO, 80309
Martha.Palmer@colorado.edu

Abstract

This paper shows that two uncertainty-based active learning methods, combined with a maximum entropy model, work well on learning English verb senses. Data analysis on the learning process, based on both instance and feature levels, suggests that a careful treatment of feature extraction is important for the active learning to be useful for WSD. The overfitting phenomena that occurred during the active learning process are identified as classic overfitting in machine learning based on the data analysis.

1 Introduction

Corpus-based methods for word sense disambiguation (WSD) have gained popularity in recent years. As evidenced by the SENSEVAL exercises (http://www.senseval.org), machine learning models supervised by sense-tagged training corpora tend to perform better on the lexical sample tasks than unsupervised methods. However, WSD tasks typically have very limited amounts of training data due to the fact that creating large-scale high-quality sense-tagged corpora is difficult and time-consuming. Therefore, the lack of sufficient labeled training data has become a major hurdle to improving the performance of supervised WSD.

A promising method for solving this problem could be the use of active learning. Researchers use active learning methods to minimize the labeling of examples by human annotators. A decrease in overall labeling occurs because active learners (the machine learning models used in active learning) pick more informative examples for the target word (a word whose senses need to be learned) than those that would be picked randomly. Active learning requires human labeling of the newly selected training data to ensure high quality.

We focus here on *pool-based* active learning where there is an abundant supply of unlabeled data, but where the labeling process is expensive. In NLP problems such as text classification (Lewis and Gale, 1994; McCallum and Nigam, 1998), statistical parsing (Tang *et al.*, 2002), information extraction (Thompson *et al.*, 1999), and named entity recognition (Shen *et al.*, 2004), pool-based active learning has produced promising results.

This paper presents our experiments in applying two active learning methods, a min-margin based method and a Shannon-entropy based one, to the task of the disambiguation of English verb senses. The contribution of our work is not only in demonstrating that these methods work well for the active learning of coarse-grained verb senses, but also analyzing the behavior of the active learning process on two levels: the instance level and the feature level. The analysis suggests that a careful treatment of feature design and feature generation is important for a successful application of active learning to WSD. We also accounted for the overfitting phenomena that occurred in the learning process based on our data analysis.

The rest of the paper is organized as follows. In Section 2, we introduce two uncertainty sampling methods used in our active learning experiments and review related work in using active learning for WSD. We then present our active learning experiments on coarse-grained English verb senses in Section 3 and analyze the active learning

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 120–127,
New York, June 2006.

process in Section 4. Section 5 presents conclusions of our study.

2 Active Learning Algorithms

The methods evaluated in this work fit into a common framework described by Algorithm 1 (see Table 1). The key difference between alternative active learning methods is how they assess the value of labeling individual examples, i.e., the methods they use for ranking and selecting the candidate examples for labeling. The framework is wide open to the type of ranking rule employed. Usually, the ranking rule incorporates the model trained on the currently labeled data. This is the reason for the requirement of a partial training set when the algorithm begins.

Algorithm 1
Require: initial training set, pool of unlabeled examples
Repeat
Select *T* random examples from pool
Rank *T* examples according to active learning rule
Present the top-ranked example to oracle for labeling
Augment the training set with the new example
Until Training set reaches desirable size

Table 1. A Generalized Active Learning Loop

In our experiments we look at two variants of the *uncertainty sampling* heuristic: entropy sampling and margin sampling. Uncertainty sampling is a term invented by Lewis and Gale (Lewis and Gale, 1994) to describe a heuristic where a probabilistic classifier picks examples for which the model's current predictions are least certain. The intuitive justification for this approach is that regions where the model is uncertain indicate a decision boundary, and clarifying the position of decision boundaries is the goal of learning classifiers. Schein (2005) demonstrates the two methods run quickly and compete favorably against alternatives when combined with the logistic regression classifier.

2.1 Entropy Sampling

A key question is how to measure uncertainty. Different methods of measuring uncertainty will lead to different variants of uncertainty sampling. We will look at two such measures. As a convenient notation we use **q** (a vector) to represent the trained model's predictions, with q_c equal to the predicted probability of class c. One method is to pick the example whose prediction vector **q** displays the greatest Shannon entropy:

$$-\sum_c q_c \log q_c \quad (1)$$

Such a rule means ranking candidate examples in Algorithm 1 by Equation 1.

2.2 Margin Sampling

An alternative method picks the example with the smallest margin: the difference between the largest two values in the vector **q** (Abe and Mamitsuka, 1998). In other words, if c and c' are the two most likely categories for example x_n, the margin is measured as follows:

$$M_n = \left|\Pr(c \mid x_n) - \Pr(c' \mid x_n)\right| \quad (2)$$

In this case Algorithm 1 would rank examples by increasing values of margin, with the smallest value at the top of the ranking.

Using either method of uncertainty sampling, the computational cost of picking an example from *T* candidates is: *O(TD)* where *D* is the number of model parameters.

2.3 Related Work

To our best knowledge, there have been very few attempts to apply active learning to WSD in the literature (Fujii and Inui, 1999; Chklovski and Mihalcea, 2002; Dang, 2004). Fujii and Inui (1999) developed an example sampling method for their example-based WSD system in the active learning of verb senses in a pool-based setting. Unlike the uncertainty sampling methods (such as the two methods we used), their method did not select examples for which the system had the minimal certainty. Rather, it selected the examples such that after training using those examples the system would be most certain about its predictions on the rest of the unlabeled examples in the next iteration. This sample selection criterion was enforced by calculating a training utility function. The method performed well on the active learning of Japanese verb senses. However, the efficient computation of the training utility function relied on the nature of the example-based learning method, which made their example sampling method difficult to export to other types of machine learning models.

Open Mind Word Expert (Chklovski and Mihalcea, 2002) was a real application of active learning for WSD. It collected sense-annotated examples from the general public through the Web to create the training data for the SENSEVAL-3 lexical sample tasks. The system used the

disagreement of two classifiers (which employed different sets of features) on sense labels to evaluate the difficulty of the unlabeled examples and ask the web users to tag the difficult examples it selected. There was no formal evaluation for this active learning system.

Dang (2004) used an uncertainty sampling method to get additional training data for her WSD system. At each iteration the system selected a small set of examples for which it had the lowest confidence and asked the human annotators to tag these examples. The experimental results on 5 English verbs with fine-grained senses (from WordNet 1.7) were a little surprising in that active learning performed no better than random sampling. The proposed explanation was that the quality of the manually sense-tagged data was limited by an inconsistent or unclear sense inventory for the fine-grained senses.

3 Active Learning Experiments

3.1 Experimental Setting

We experimented with the two uncertainty sampling methods on 5 English verbs that had coarse-grained senses (see Table 2), as described below. By using coarse-grained senses, we limit the impact of noisy data due to unclear sense boundaries and therefore can get a clearer observation of the effects of the active learning methods themselves.

verb	# of sen.	baseline acc. (%)	Size of data for active learning	Size of test data
Add	3	91.4	400	100
Do	7	76.9	500	200
Feel	3	83.6	400	90
See	7	59.7	500	200
Work	9	68.3	400	150

Table 2. The number of senses, the baseline accuracy, the number of instances used for active learning and for held-out evaluation for each verb

The coarse-grained senses are produced by grouping together the original WordNet senses using syntactic and semantic criteria (Palmer *et al.*, 2006). Double-blind tagging is applied to 50 instances of the target word. If the ITA < 90%, the sense entry is revised by adding examples and explanations of distinguishing criteria.

Table 2 summarizes the statistics of the data. The baseline accuracy was computed by using the "most frequent sense" heuristic to assign sense labels to verb instances (examples). The data used in active learning (Column 4 in Table 2) include two parts: an initial labeled training set and a pool of unlabeled training data. We experimented with sizes 20, 50 and 100 for the initial training set. The pool of unlabeled data had actually been annotated in advance, as in most pool-based active learning experiments. Each time an example was selected from the pool by the active learner, its label was returned to the learner. This simulates the process of asking human annotators to tag the selected unlabeled example at each time. The advantage of using such a simulation is that we can experiment with different settings (different sizes of the initial training set and different sampling methods).

The data sets used for active learning and for held-out evaluation were randomly sampled from a large data pool for each round of the active learning experiment. We ran ten rounds of the experiments for each verb and averaged the learning curves for the ten rounds.

In the experiments, we used random sampling (picking up an unlabeled example randomly at each time) as a lower bound. Another control (ultimate-maxent) was the learner's performance on the test set when it was trained on a set of labeled data that were randomly sampled from a large data pool and equaled the amount of data used in the whole active learning process (e.g., 400 training data for the verb *add*).

The machine learning model we used for active learning was a regularized maximum entropy (MaxEnt) model (McCallum, 2002). The features used for disambiguating the verb senses included topical, collocation, syntactic (e.g., the subject, object, and preposition phrases taken by a target verb), and semantic (e.g., the WordNet synsets and hypernyms of the head nouns of a verb's NP arguments) features (Chen and Palmer, 2005).

3.2 Experimental Results

Due to space limits, Figure 1 only shows the learning curves for 4 verbs *do, feel, see,* and *work* (size of the initial training set = 20). The curve for the verb *add* is similar to that for *feel*. These curves clearly show that the two uncertainty sampling methods, the entropy-based (called entropy-maxent in the figure) and the margin-based (called min_margin-maxent), work very well for active learning of the senses of these verbs.

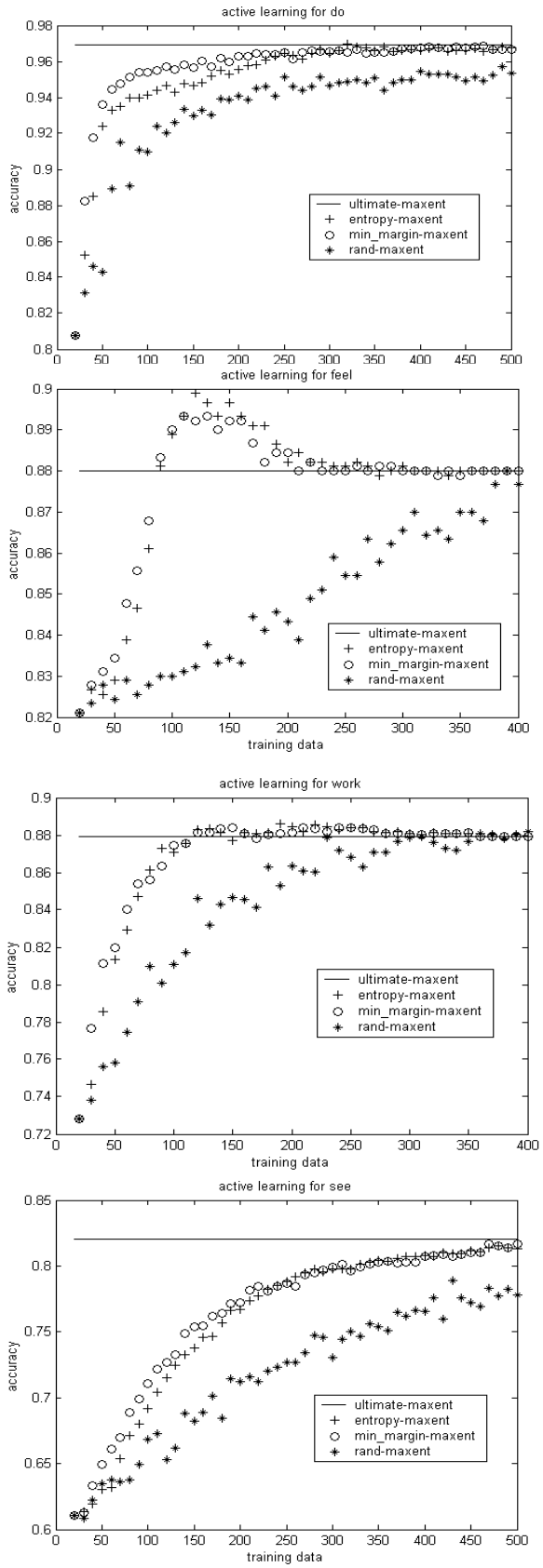


Figure 1 Active learning for four verbs

Both methods outperformed the random sampling method in that they reached the upper-bound accuracy earlier and had smoother learning curves. For the four verbs *add, do, feel* and *see*, their learning curves reached the upper bound at about 200~300 iterations, which means 1/2 or 1/3 of the annotation effort can be saved for these verbs by using active learning, while still achieving the same level of performance as supervised WSD without using active learning. Given the large-scale annotation effort currently underway in the OntoNotes project (Hovy *et al.*, 2006), this could provide considerable savings in annotation effort and speed up the process of providing sufficient data for a large vocabulary. The OntoNotes project has now provided coarse-grained entries for over 350 verbs, with corresponding double–blind annotation and adjudication in progress. As this adjudicated data becomes available, we will be able to train our system accordingly. Preliminary results for 22 of these coarse-grained verbs (with an average grouping polysemy of 4.5) give us an average accuracy of 86.3%. This will also provide opportunities for more experiments with active learning, where there are enough instances. Active learning could also be beneficial in porting these supervised taggers to new genres with different sense distributions.

We also experimented with different sizes of the initial training set (20, 50 and 100) and found no significant differences in the performance at different settings. That means, for these 5 verbs, only 20 labeled training instances will be enough to initiate an efficient active learning process.

From Figure 1, we can see that the two uncertainty sampling methods generally perform equally well except that for the verb *do*, the min-margin method is slightly better than the entropy method at the beginning of active learning. This may not be so surprising, considering that the two methods are equal for two-class classification tasks (see Equations 1 and 2 for their definition) and the verbs used in our experiments have coarse-grained senses and often have only 2 or 3 major senses.

An interesting phenomenon observed from these learning curves is that for the two verbs *add* and *feel*, the active learner reached the upper bound very soon (at about 100 iterations) and then even breached the upper bound. However, when the training set was extended, the learner's performance dropped and eventually returned to

the same level of the upper bound. We discuss the phenomenon below.

4 Analysis of the Learning Process

In addition to verifying the usefulness of active learning for WSD, we are also interested in a deeper analysis of the learning process. For example, why does the active learner's performance drop sometimes during the learning process? What are the characteristics of beneficial features that help to boost the learner's accuracy? How do we account for the overfitting phenomena that occurred during the active learning for the verbs *add* and *feel*? We analyzed the effect of both instances and features throughout the course of active learning using min-margin-based sampling.

4.1 Instance-level Analysis

Intuitively, if the learner's performance drops after a new example is added to the training set, it is likely that something has gone wrong with the new example. To find out such *bad* examples, we define a measure *credit_inst* for instance *i* as:

$$\frac{1}{m}\sum_{r=1}^{m}\sum_{l=1}^{n} sel(i,l)(Acc_{l+1} - Acc_l) \qquad (3)$$

where Acc_l and Acc_{l+1} are the classification accuracies of the active learner at the *lth* and *(l+1)th* iterations. *n* is the total number of iterations of active learning and *m* is the number of rounds of active learning (*m*=10 in our case). $sel(i,l)$ is 1 *iff* instance *i* is selected by the active learner at the *l*th iteration and is 0 if otherwise.

An example is a *bad example* if and only if it satisfies the following conditions:

a) its *credit_inst* value is negative

b) it increases the learner's performance, if it does, less often than it decreases the performance in the 10 rounds.

We ranked the bad examples by their *credit_inst* values and their frequency of decreasing the learner's performance in the 10 rounds. Table 3 shows the top five bad examples for *feel* and *work*. There are several reasons why the bad examples may hurt the learner's performance. Column 3 of Table 3 proposes reasons for many of our bad examples. We categorized these reasons into three major types.

I. The major senses of a target verb depend heavily on the semantic categories of its NP arguments but WordNet sometimes fails to provide the appropriate semantic categories (features) for the head nouns of these NP arguments. For example, *feel* in *the board apparently felt no pressure* has Sense 1 (experience). In Sense 1, *feel* typically takes an ***animate*** subject. However, *board*, the head word of the verb's subject in the above sentence has no animate meanings defined in WordNet. Even worse, the major meaning of *board*, i.e., *artifact*, is typical for the subject of *feel* in Sense 2 (touch, grope). Similar semantic type mismatches hold for the last four bad examples of the verb *work* in Table 3.

II. The contexts of the target verb are difficult for our feature exaction module to analyze. For example, the antecedent for the pronoun subject *they* in the first example of *work* in Table 3 should be *ringers*, an ***agent*** subject that is typical for Sense 1 (exert oneself in an activity). However, the feature exaction module found the wrong antecedent *changes* that is an unlikely fit for the intended verb sense. In the fourth example for *feel*, the feature extraction module cannot handle the expletive "it" (a dummy subject) in "it was felt that", therefore, it cannot identify the typical syntactic pattern for Sense 3 (find, conclude), i.e., *subject+feel+relative clause*.

III. Sometimes, deep semantic and discourse analyses are needed to get the correct meaning of the target verb. For example, in the third example of *feel*, "…, *he or she feels age creeping up*", it is difficult to tell whether the verb has Sense 1 (experience) or Sense 3 (find) without an understanding of the meaning of the relative clause and without looking at a broader discourse context. The syntactic pattern identified by our feature extraction module, *subject+feel+relative clause*, favors Sense 3 (find), which leads to an inaccurate interpretation for this case.

Recall that the motivation behind uncertainty samplers is to find examples near decision boundaries and use them to clarify the position of these boundaries. Active learning often does find informative examples, either ones from the less common senses or ones close to the boundary between the different senses. However, active learning also identifies example sentences that are difficult to analyze. The failure of our feature extraction module, the lack of appropriate semantic categories for certain NP arguments in WordNet, the lack of deep analysis (semantic and discourse analysis) of the context of the target verb can all

feel	**Proposed reasons for bad examples**	**Senses**
Some days the coaches make you feel as though you are part of a large herd of animals .	?	S1: experience
And , with no other offers on the table , the board apparently felt no pressure to act on it.	subject: *board*, no "animate" meaning in WordNet	S1: experience
Sometimes a burst of aggressiveness will sweep over a man -- or his wife -- because he or she feels age creeping up.	syntactic pattern: sbj+*fee*l+relative clause headed by *that*, a typical pattern for Sense 3 (find) rather than Sense 1 (experience)	S1: experience
At this stage it was felt I was perhaps more pertinent as chief. executive .	syntactic pattern: sbj+*fee*l+relative clause, typical for Sense 3 (find) but has not been detected by the feature exaction module	S3: find, conclude
I felt better Tuesday evening when I woke up.	?	S1: experience
Work		
When their changes are completed, and after they have worked up a sweat, ringers often ……	subject: *they*, the feature exaction module found the wrong antecedent (*changes* rather than *ringers*) for *they*	S1: exert oneself in an activity
Others grab books, records , photo albums , sofas and chairs , working frantically in the fear that an aftershock will jolt the house again .	subject: *others* (means *people* here), no definition in WordNet	S1: exert oneself in an activity
Security Pacific 's factoring business works with companies in the apparel, textile and food industries …	subject: *business*, no "animate" meaning in WordNet	S1: exert oneself in an activity
… ; blacks could work there , but they had to leave at night .	subject: *blacks*, no "animate" meaning in WordNet	S1: exert oneself in an activity
… has been replaced by alginates (gelatin-like material) that work quickly and accurately and with least discomfort to a child .	subject: *alginates*, unknown by WordNet	S2: perform, function, behave

Table 3 Data analysis of the top-ranked bad examples found for two verbs

produce misleading features. Therefore, in order to make active learning useful for its applications, both identifying difficult examples *and* getting good features for these examples are equally important. In other words, a careful treatment of feature design and feature generation is necessary for a successful application of active learning.

There is a positive side to identifying such "bad" examples; one can have human annotators look at the features generated from the sentences (as we did above), and use this to improve the data or the classifier. Note that this is exactly what we did above: the identification of bad sentences was automatic, and they could then be reannotated or removed from the training set or the feature extraction module needs to be refined to generate informative features for these sentences.

Not all sentences have obvious interpretations; hence the two question marks in Table 3. An example can be bad for many reasons: conflicting features (indicative of different senses), misleading features (indicative of non-intended senses), or just containing random features that are incorrectly incorporated into the model. We will return to this point in our discussion of the overfitting phenomena for active learning in Section 4.3.

4.2 Feature-level Analysis

The purpose of our feature-level analysis is to identify informative features for verb senses. The learning curve of the active learner may provide some clues. The basic idea is, if the learner's performance increases after adding a new example, it is likely that the *good* example contains good features that contribute to the clarification of sense boundaries. However, the feature-level analysis is much less straightforward than the instance-level analysis since we cannot simply say the features that are active (present) in this *good* example are all good. Rather, an example often contains both good and bad features, and many other features that are somehow neutral or uninformative. The interaction or balance between these features determines the final outcome. On the other hand, a statistics based analysis may help us to find features that tend to be good or bad. For this analysis, we define a measure *credit_feat* for feature *i* as:

$$\frac{1}{m}\sum_{r=1}^{m}\sum_{l=1}^{n} active(i,l)(Acc_{l+1} - Acc_l)\frac{1}{act_l} \qquad (4)$$

where $active(i,l)$ is 1 iff feature i is active in the example selected by the active learner at the lth iteration and is 0 if otherwise. act_l is the total number of active features in the example selected at the lth iteration. n and m have the same definition as in Equation 3.

A feature is regarded as *good* if its *credit_feat* value is positive. We ranked the good features by their *credit_feat* values. By looking at the top-ranked good features for the verb *work* (due to space limitations, we omit the table data), we identify two types of typically good features.

The first type of good feature occurs frequently in the data and has a frequency distribution over the senses similar to the data distribution over the senses. Such features include those denoting that the target verb takes a subject (*subj*), is not used in a passive mode (*morph_normal*), does not take a direct object (*intransitive*), occurs in present tense (*word_work, pos_vb, word_works, pos_vbz*), and semantic features denoting an ***abstract*** subject (subjsyn_16993 [1]) or an ***entity*** subject (subjsyn_1742), *etc.* We call such features *background* features. They help the machine learning model learn the appropriate sense distribution of the data. In other words, a learning model only using such features will be equal to the "most frequent sense" heuristic used in WSD.

Another type of good feature occurs less frequently and has a frequency distribution over senses that mismatches with the sense distribution of the data. Such features include those denoting that the target verb takes an inanimate subject (*subj_it*), takes a particle *out* (*prt_out*), is followed directly by the word *out* (word+1_out), or occurs at the end of the sentence. Such features are indicative of less frequent verb senses that still occur fairly frequently in the data. For example, taking an ***inanimate*** subject (*subj_it*) is a strong clue for Sense 2 (perform, function, behave) of the verb *work*. Occurring at the end of the sentence is also indicative of Sense 2 since when *work* is used in Sense 1 (exert oneself in an activity), it tends to take adjuncts to modify the activity as in *He is working hard to bring up his grade*.

[1] Those features are from the WordNet. The numbers are WordNet ids of synsets and hypernyms.

There are some features that don't fall into the above two categories, such as the topical feature *tp_know* and the collocation feature *pos-2_nn*. There are no obvious reasons why they are good for the learning process, although it is possible that the combination of two or more such features could make a clear sense distinction. However, this hypothesis cannot be verified by our current statistics-based analysis. It is also worth noting that our current feature analysis is post-experimental (i.e., based on the results). In the future, we will try automatic feature selection methods that can be used in the training phase to select useful features and/or their combinations.

We have similar results for the feature analysis of the other four verbs.

4.3 Account for the Overfitting Phenomena

Recall that in the instance-level analysis in Section 4.1, we found that some examples hurt the learning performance during active learning but for no obvious reasons (the two examples marked by ? in Table 3). We found that these two examples occurred in the overfitting region for *feel*. By looking at the bad examples (using the same definition for *bad* example as in Section 4.1) that occurred in the overfitting region for both *feel* and *add*, we identified two major properties of these examples. First, most of them occurred only once as bad examples (19 out 23 for *add* and 40 out of 63 for *feel*). Second, many of the examples had no obvious reasons for their badness.

Based on the above observations, we believe that the overfitting phenomena that occurred for the two verbs during active learning is typical of classic overfitting, which is consistent with a *"death by a thousand mosquito bites"* of rare bad features, and consistent with there often being (to mix a metaphor) no *"smoking gun"* of a bad feature/instance that is added in, especially in the region far away from the starting point of active learning.

5 Conclusions

We have shown that active learning can lead to substantial reductions (often by half) in the number of observations that need to be labeled to achieve a given accuracy in word sense disambiguation, compared to labeling randomly selected instances. In a follow-up experiment, we also compared a larger number of different active learning methods.

The results suggest that for tasks like word sense disambiguation where maximum entropy methods are used as the base learning models, the minimum margin active criterion for active learning gives superior results to more comprehensive competitors including bagging and two variants of query by committee (Schein, 2005). By also taking into account the high running efficiency of the min-margin method, it is a very promising active learning method for WSD.

We did an analysis on the learning process on two levels: instance-level and feature-level. The analysis suggests that a careful treatment of feature design and feature generation is very important for the active learner to take advantage of the difficult examples it finds during the learning process. The feature-level analysis identifies some characteristics of good features. It is worth noting that the good features identified are not particularly tied to active learning, and could also be obtained by a more standard feature selection method rather than by looking at how the features provide benefits as they are added in.

For a couple of the verbs examined, we found that active learning gives higher prediction accuracy midway through the training than one gets after training on the entire corpus. Analysis suggests that this is not due to *bad* examples being added to the training set. It appears that the widely used maximum entropy model with Gaussian priors is overfitting: the model by including too many features and thus fitting noise as well as signal. Using different strengths of the Gaussian prior does not solve the problem. If a very strong prior is used, then poorer accuracy is obtained. We believe that using appropriate feature selection would cause the phenomenon to vanish.

Acknowledgements

This work was supported by National Science Foundation Grant NSF-0415923, Word Sense Disambiguation, the DTO-AQUAINT NBCHC-040036 grant under the University of Illinois subcontract to University of Pennsylvania 2003-07911-01 and the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-C-0022. 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 National Science Foundation, the DTO, or DARPA.

References

Naoki Abe and Hiroshi Mamitsuka. 1998. Query learning strategies using boosting and bagging. In *Proc. of ICML1998*, pages 1–10.

Jinying Chen and Martha Palmer. 2005. Towards Robust High Performance Word Sense Disambiguation of English Verbs Using Rich Linguistic Features, In *Proc. of IJCNLP2005*, Oct., Jeju, Republic of Korea.

Tim Chklovski and Rada Mihalcea, Building a Sense Tagged Corpus with Open Mind Word Expert, in *Proceedings of the ACL 2002 Workshop on "Word Sense Disambiguation: Recent Successes and Future Directions"*, Philadelphia, July 2002.

Hoa T. Dang. 2004. Investigations into the role of lexical semantics in word sense disambiguation. PhD Thesis. University of Pennsylvania.

Atsushi Fujii, Takenobu Tokunaga, Kentaro Inui, Hozumi Tanaka. 1998. Selective sampling for example-based word sense disambiguation, Computational Linguistics, v.24 n.4, p.573-597, Dec.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw and Ralph Weischedel. OntoNotes: The 90% Solution. Accepted by *HLT-NAACL06*. Short paper.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In W. Bruce Croft and Cornelis J. van Rijsbergen, editors, *Proceedings of SIGIR-94*, Dublin, IE.

Andrew K. McCallum. 2002. MALLET: A Machine Learning for Language Toolkit. *http://www.cs.umass.edu/~mccallum/mallet*.

Andew McCallum and Kamal Nigam. 1998. Employing EM in pool-based active learning for text classification. In *Proc. of ICML '98*.

Martha Palmer, Hoa Trang Dang and Christiane Fellbaum. (to appear, 2006). Making fine-grained and coarse-grained sense distinctions, both manually and automatically. *Natural Language Engineering*.

Andrew I. Schein. 2005. Active Learning for Logistic Regression. Ph.D. Thesis. Univ. of Pennsylvania.

Dan Shen, Jie Zhang, Jian Su, Guodong Zhou and Chew Lim Tan. 2004 Multi-criteria-based active learning for named entity recognition, In *Proc. of ACL04*, Barcelona, Spain.

Min Tang, Xiaoqiang Luo, and Salim Roukos. 2002. Active learning for statistical natural language parsing. In *Proc. of ACL* 2002.

Cynthia A. Thompson, Mary Elaine Califf, and Raymond J. Mooney. 1999. Active learning for natural language parsing and information extraction. In *Proc. of ICML-99*.

Unknown word sense detection as outlier detection

Katrin Erk
Computational Linguistics
Saarland University
Saarbrücken, Germany
erk@coli.uni-sb.de

Abstract

We address the problem of *unknown word sense detection*: the identification of corpus occurrences that are not covered by a given sense inventory. We model this as an instance of *outlier detection*, using a simple nearest neighbor-based approach to measuring the resemblance of a new item to a training set. In combination with a method that alleviates data sparseness by sharing training data across lemmas, the approach achieves a precision of 0.77 and recall of 0.82.

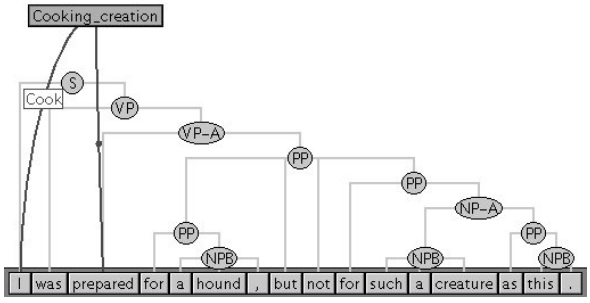


Figure 1: Wrong assignment due to missing sense: from the Hound of the Baskervilles, Ch. 14

1 Introduction

If a system has seen only positive examples, how does it recognize a negative example? This is the problem addressed by *outlier detection*, also called *novelty detection*[1] (Markou and Singh, 2003a; Markou and Singh, 2003b; Marsland, 2003): to detect novel or unknown items that differ from all the seen training data. Outlier detection approaches typically derive some model of "normal" objects from the training set and use a distance measure and a threshold to detect abnormal items.

In this paper, we apply outlier detection techniques to the task of *unknown sense detection*: the identification of corpus occurrences that are not covered by a given sense inventory. The training set against which new occurrences are compared will consist of sense-annotated text.

Unknown sense detection is related to word sense disambiguation (WSD) and to word sense discrimination (Schütze, 1998), but differs from both. In WSD all senses are assumed known, and the task is to select one of them, while in unknown sense detection the task is to decide whether a given occurrence matches any of the known senses or none of them, and all training instances, regardless of the sense to which they belong, are modeled as *one* group of *known* data. Unknown sense detection also differs from word sense discrimination, where no sense inventory is given and the task is to group occurrences into senses. In unknown sense detection the model respects the given word senses.

The main motivation for this study comes from *shallow semantic parsing*, by which we mean a combination of WSD and the automatic assignment of

[1]The term *novelty detection* is also used for the distinction of novel and repeated information in information retrieval, a different if related topic.

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 128–135,
New York, June 2006.

semantic roles to free text. In cases where a sense is missing from the inventory, WSD will wrongly assign one of the existing senses. Figure 1 shows an example, a sentence from the *Hound of the Baskervilles*, analyzed by the SHALMANESER (Erk and Pado, 2006) shallow semantic parser. The analysis is based on FrameNet (Baker et al., 1998), a resource that lists senses and semantic roles for English expressions. FrameNet is lacking a sense of "expectation" or "being mentally prepared" for the verb *prepare*, so *prepared* has been assigned the sense COOKING_CREATION, a possible but improbable analysis[2]. Such erroneous labels can be fatal when further processing builds on the results of shallow semantic parsing, e.g. for drawing inferences. Unknown sense detection can prevent such mistakes.

All sense inventories face the problem of missing senses, either because of their small overall size (as is the case for some non-English WordNets) or when they encounter domain-specific senses. Our study will be evaluated on FrameNet because of our main aim of improving shallow semantic parsing, but the method we propose is applicable to any sense inventory that has annotated data; in particular, it is also applicable to WordNet.

In this paper we model unknown sense detection as outlier detection, using a simple Nearest Neighbor-based method (Tax and Duin, 2000) that compares the local probability density at each test item with that of its nearest training item.

To our knowledge, there exists no other approach to date to the problem of detecting unknown senses. There are, however, approaches to the complementary problem of determining the closest known sense for unknown words (Widdows, 2003; Curran, 2005; Burchardt et al., 2005), which can be viewed as the logical next step after unknown sense detection.

Plan of the paper. After a brief sketch of FrameNet in Section 2, we describe the experimental setup used throughout this paper in Section 3. Section 4 tests whether a very simple model suffices for detecting unknown senses: a threshold on confidence scores returned by the SHALMANESER WSD system. The result is that recall is much too low. Section 5 introduces the NN-based outlier detection approach that we use in section 6 for unknown sense detection, with better results than in the first experiment but still low recall. Section 7 repeats the experiment of section 6 with added training data, making use of the fact that one semantic class in FrameNet typically pertains to several lemmas and achieving a marked improvement in results.

[2]Unfortunately, the semantic roles have been mis-assigned by the system. The word *I* should fill the FOOD role, while *for a hound* could be assigned the optional RECEIVER role.

2 FrameNet

Frame Semantics (Fillmore, 1982) models the meanings of a word or expression by reference to *frames* which describe the background and situational knowledge necessary for understanding what the predicate is "about". Each frame provides its specific set of semantic roles.

The Berkeley FrameNet project (Baker et al., 1998) is building a semantic lexicon for English describing the frames and linking them to the words and expressions that can *evoke* them. These can be verbs as well as nouns, adjectives, prepositions, adverbs, and multiword expressions. Frames are linked by IS-A and other relations. Currently, FrameNet contains 609 frames with 8,755 lemma-frame pairs, of which 5,308 are exemplified in annotated sentences from the British National Corpus. The annotation comprises 133,846 sentences.

As FrameNet is a growing resource, many lemmas are still lacking senses, and many senses are still lacking annotation. This is problematic for the use of FrameNet analyses as a basis for inferences over text, as e.g. in Tatu and Moldovan (2005).

For example, the verb *prepare* from Figure 1 is associated with the frames

COOKING_CREATION: prepare food
ACTIVITY_PREPARE: get ready for an activity
ACTIVITY_READY_STATE: be ready for an activity
WILLINGNESS: be willing

of which only the COOKING_CREATION sense has been annotated. The sense in Figure 1 is not covered yet: ACTIVITY_READY_STATE would be more appropriate than COOKING_CREATION, but still not optimal, since the sentence refers to a mental state rather than the preparation of an activity.

3 Experimental setup and data

Experimental setup. To evaluate an unknown sense detection system, we need occurrences that are guaranteed not to belong to any of the seen senses. To that end we use sense-annotated data, in our case the FrameNet annotated sentences, simulating unknown senses by designating one sense of each ambiguous lemma as unknown. All occurrences of that sense are placed in the test set, while occurrences of all other senses are split randomly between training and test set, using 5-fold cross-validation. We repeat the experiment with each of the senses of an ambiguous lemma playing the part of the unknown sense once. Viewing each cross-validation run for each unknown sense as a separate experiment, we then report precision and recall averaged over unknown senses and cross-validation runs.

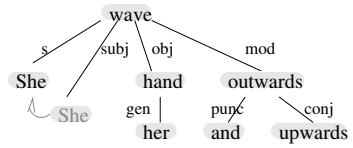
It may seem questionable that in this experimental setup, the *unknown sense* occurrences of each lemma all belong to the same sense. However, this does not bias the experiment since none of the models we study take advantage of the shape of the test set in any way. Rather, each test item is classified individually, without recourse to the other test items.

Data. All experiments in this paper were performed on the FrameNet 1.2 annotated data pertaining to ambiguous lemmas. After removal of instances that were annotated with more than one sense, we obtain 26,496 annotated sentences for the 1,031 ambiguous lemmas. They were parsed with Minipar (Lin, 1993); named entities were computed using Heart of Gold (Callmeier et al., 2004).

4 Experiment 1: WSD confidence scores for unknown sense detection

In this section we test a very simple model of unknown sense detection: Classifiers often return a confidence score along with the assigned label. We will try to detect unknown senses by a threshold on confidence scores, declaring anything below the threshold as unknown. Note that this method can only be applied to lemmas that have more than one sense, since for single-sense lemmas the system will always return the maximum confidence score.

Data. While the approach that we follow in this section is applicable to all lemmas with at least two senses, we need lemmas with at least three senses to evaluate it: One of the senses of each lemma is treated as *unknown*, which for lemmas with three or more senses leaves at least two senses for the training set. This reduces our data set to 125 lemmas with 7,435 annotated sentences.



(1): `subj`, `obj`, `mod` (since `s` and `subj` corefer, we use only one of them)
(2): `she`, `hand`, `outwards`
(3): `subj-she`, `obj-hand`, `mod-outwards`
(4): `mod-obj-subj`

Figure 2: Sample Minipar parse and extracted grammatical function features

Modeling. We test whether the WSD system built into SHALMANESER (Erk, 2005) can distinguish *known sense* items from *unknown sense* items reliably by its confidence scores. The system extracts a rich feature set, which forms the basis of all three experiments in this paper:

- a bag-of-words context, with a window size of one sentence;
- bi- and trigrams centered on the target word;
- grammatical function information: for each dependent of the target, (1) its function label, (2) its headword, and (3) a combination of both are used as features. (4) The concatenation of all function labels constitutes another feature. For PPs, function labels are extended by the preposition. As an example, Figure 2 shows a BNC sentence and its grammatical function features.
- for verb targets, the target voice.

The feature set is based on Florian et al. (2002) but contains additional syntax-related features. Each word-related feature is represented as four features for word, lemma, part of speech, and named entity.

SHALMANESER trains one Naive Bayes classifier per lemma to be disambiguated. For this experiment,

θ	Precision		Recall	
0.5	0.6524	(σ 0.115)	0.0011	(σ 0.0004)
0.75	0.7855	(σ 0.0086)	0.0527	(σ 0.0013)
0.9	0.7855	(σ 0.0093)	0.1006	(σ 0.0021)
0.98	0.7847	(σ 0.0073)	0.1744	(σ 0.0025)

Table 1: Experiment 1: Results for label *unknown sense*, WSD confidence level approach. θ: confidence threshold. σ: std. dev.

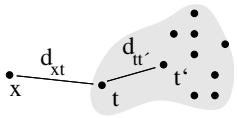


Figure 3: Outlier detection by comparing distances between nearest neighbors

all system parameters were set to their default settings. To detect unknown senses building on this WSD system, we use a fixed confidence threshold and label all items below the threshold as *unknown*.

Results and discussion. Table 1 shows precision and recall for labeling instances as *unknown* using different confidence thresholds θ, averaged over unknown senses and 5-fold cross-validation[3]. We see that while the precision of this method is acceptable at 0.74 to 0.765, recall is extremely low, i.e. almost no items were labeled *unknown*, even at a threshold of 0.98. However, SHALMANESER has very high confidence values overall: Only 14.5% of all instances in this study had a confidence value of 0.98 or below (7,697 of 53,206).

We conclude that with the given WSD system and (rather standard) features, this simple method cannot detect items with an unknown sense reliably. This may be due to the indiscriminately high confidence scores; or it could indicate that classifiers, which are geared at *distinguishing* between known classes rather than *detecting* objects that differ from all seen data, are not optimally suited to the task. However, one further disadvantage of this approach is that, as mentioned above, it can only be applied to lemmas with more than one annotated sense. For FrameNet 1.2, this comprises only 19% of the lemmas.

5 A nearest neighbor-based method for outlier detection

In the previous section we have tested a simple approach to unknown sense detection using WSD confidence scores. Our conclusion was that it was not a viable approach, given its low recall and given that it is only applicable to lemmas with more than one known sense. In this section we introduce an alternative approach, which uses distances to nearest neighbors to detect outliers.

In general, the task of outlier detection is to decide whether a new object belongs to a given training set or not. Typically, outlier detection approaches derive some boundary around the training set, or they derive from the set some model of "normality" to which new objects are compared (Markou and Singh, 2003a; Markou and Singh, 2003b; Marsland, 2003). Applications of outlier detection include fault detection (Hickinbotham and Austin, 2000), hand writing deciphering (Tax and Duin, 1998; Schölkopf et al., 2000), and network intrusion detection (Yeung and Chow, 2002; Dasgupta and Forrest, 1999). One standard approach to outlier detection estimates the probability density of the training set, such that a test object can be classified as an outlier or non-outlier according to its probability of belonging to the set.

Rather than estimating the complete density function, Tax and Duin (2000) approximate local density at the test object by comparing distances between nearest neighbors. Given a test object x, the approach considers the training object t nearest to x and compares the distance d_{xt} between x and t to the distance $d_{tt'}$ between t and its own nearest training data neighbor t'. Then the quotient between the distances is used as an indicator of the (ab-)normality of the test object x:

$$p_{NN}(x) = \frac{d_{xt}}{d_{tt'}}$$

When the distance d_{xt} is much larger than $d_{tt'}$, x is considered an outlier. Figure 3 illustrates the idea.

The normality or abnormality of test objects is decided by a fixed threshold θ on p_{NN}. The lowest

[3]Note that the minimum confidence score is 0.5 if 2 senses are present in the training set, 0.33 for 3 present senses etc.

threshold that makes sense is 1.0, which rejects any x that is further apart from its nearest training neighbor t than t is from its neighbor. Tax and Duin use Euclidean distance, i.e.

$$d_{xt} = \sqrt{\sum_i (x_i - t_i)^2}$$

Applied to feature vectors with entries either 0 or 1, this corresponds to the size of the symmetric difference of the two feature sets.

6 Experiment 2: NN-based outlier detection

In this section we use the NN-based outlier detection approach of the previous section for an experiment in unknown sense detection. Experimental setup and data are as described in Section 3.

Modeling. We model unknown sense detection as an outlier detection task, using Tax and Duin's outlier detection approach that we have outlined in the previous section. Nearest neighbors (by Euclidean distance) were computed using the ANN tool (Mount and Arya, 2005). We compute one outlier detection model per lemma. With training and test sets constructed as described in Section 3, the average training set comprises 22.5 sentences.

We use the same features as in Section 4, with feature vector entries of 1 for present and 0 for absent features. For a more detailed analysis of the contribution of different feature types, we test on reduced as well as full feature vectors:

`All`: full feature vectors

`Cx`: only bag-of-word context features (words, lemmas, POS, NE)

`Syn`: function labels of dependents

`Syn-hw`: `Syn` plus headwords of dependents

We compare the NN-based model to that of Experiment 1, but not to any simpler baseline. While for WSD it is possible to formulate simple frequency-based methods that can serve as a baseline, this is not so in unknown sense detection because the frequency of unknown senses is, by definition, unknown. Furthermore, the number of annotated sentences per sense in FrameNet depends on the number of subcategorization frames of the lemma rather than the frequency of the sense, which makes frequency calculations meaningless.

Features	Precision		Recall	
`All`	0.7072	(σ 0.0088)	0.2683	(σ 0.0043)
`Cx`	0.7016	(σ 0.0041)	0.3511	(σ 0.0035)
`Syn`	0.8333	(σ 0.0085)	0.2099	(σ 0.0042)
`Syn-hw`	0.7784	(σ 0.0029)	0.2368	(σ 0.0022)

Table 2: Experiment 2: Results for label *unknown sense*, NN-based outlier detection, $\theta = 1.0$. σ: standard deviation

	Precision			Recall		
Features	all	≥ 10	≥ 20	all	≥ 10	≥ 20
`All`	0.71	0.70	0.67	0.27	0.35	0.45
`Cx`	0.70	0.70	0.67	0.35	0.47	0.58
`Syn`	0.83	0.81	0.77	0.21	0.22	0.21
`Syn-hw`	0.78	0.76	0.73	0.24	0.28	0.31

Table 3: Experiment 2: Results by training set size, $\theta = 1.0$

Results. Table 2 shows precision and recall for labeling instances as *unknown* using a distance quotient threshold of θ=1.0, averaged over unknown senses and over 5-fold cross-validation. We see that recall is markedly higher than in Experiment 1, especially for the two conditions that include context words, `All` and `Cx`. The syntax-based conditions `Syn` and `Syn-hw` show a higher precision, with a less pronounced increase in recall.

Raising the distance quotient threshold results in little change in precision, but a large drop in recall. For example, `All` vectors with a threshold of $\theta = 1.1$ achieve a recall of 0.14 in comparison to 0.27 for $\theta = 1.0$.

Training set size is an important factor in system results. Table 3 lists precision and recall for all training sets, for training sets of size ≥ 10, and for training sets of size ≥ 20. Especially in conditions `All` and `Cx`, recall rises steeply when we only consider cases with larger training sets. However note that precision does not rise with larger training sets, rather it shows a slight decline.

Another important factor is the number of senses that a lemma has, as the upper part of Table 7 shows. For lemmas with a higher number of senses, preci-

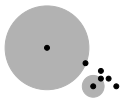


Figure 4: "Acceptance radius" of an outlier within the training set (left) and a more "normal" training set object (right)

sion is much lower, while recall is much higher.

Discussion. While results in this experiment are better than in Experiment 1 – in particular recall has risen by 19 points for `Cx` –, system performance is still not high enough to be usable in practice.

The uniformity of the training set has a large influence on performance, as Table 7 shows. The more senses a lemma has, the harder it seems to be for the model to identify *known sense* occurrences. Precision for the assignment of the *unknown* label drops, while recall rises. We see a tradeoff between precision and recall, in this table as well as in Table 3. There, we see that many more *unknown* test objects are identified when training sets are larger, but a larger training set does not translate into universally higher results.

One possible explanation for this lies in a property of Tax and Duin's approach. If a training item t is situated at distance d from its nearest neighbor in the training set, then any test item within a radius of d around t will be considered *known*. Thus we could term d the "acceptance radius" of t. Now if t is an outlier *within* the training set, then d will be large, as illustrated in Figure 4. The sparser the training set is, the more training outliers we are likely to find, with large acceptance radii that assign a label of *known* even to more distanced test items. Thus a sparse training set could lead to lower recall of *unknown sense* assignment and at the same time higher precision, as the items labeled *unknown* would be the ones at great distance from any items on the training set – conforming to the pattern in Tables 3 and 7.

7 Experiment 3: NN-based outlier detection with added training data

While the NN-based outlier detection model we used in the previous experiment showed better results than the WSD confidence model, its recall is still low. We have suggested that data sparseness may be responsible for the low performance. Consequently, we repeat the experiment of the previous section with more, but less specific, training data.

Like WordNet synsets, FrameNet frames are semantic classes that typically comprise several lemmas or expressions. So, assuming that words with similar meaning occur in similar contexts, the context features for lemmas in the same frame should be similar. Following this idea, we supplement the training data for a lemma by all the *other* annotated data for the senses that are present in the training set, where by "other data" we mean data with other target lemmas. Table 4 shows an example[4].

Target lemma: put
Senses: ENCODING, PLACING
Sense currently treated as unknown: PLACING
Extend training set by: all annotated sentences for lemmas other than *put* in the sense ENCODING: couch.v, expression.n, formulate.v, formulation.n, frame.v, phrase.v, word.v, wording.n

Table 4: Extending training sets: an example

Features	Precision		Recall	
`All`	0.7709	(σ 0.001)	0.7243	(σ 0.0018)
`Cx`	0.7727	(σ 0.0027)	0.8172	(σ 0.0035)
`Syn`	0.8571	(σ 0.0045)	0.1694	(σ 0.0012)
`Syn-hw`	0.8025	(σ 0.0041)	0.3383	(σ 0.0025)
`Syn`	0.8587	(σ 0.0081)	0.1748	(σ 0.0015)
`Syn-hw`	0.8055	(σ 0.0056)	0.3516	(σ 0.0015)

Table 5: Experiment 3: Results for label *unknown sense*, NN-based outlier detection, $\theta = 1.0$. σ: standard deviation

Modeling. Again, we use Tax and Duin's outlier detection approach for unknown sense detection. The experimental design and evaluation are the same as in Experiment 2, the only difference being the training set extension. Training set extension raises the average training set size from 22.5 to 374.

Results. Table 5 shows precision and recall for labeling instances as *unknown*, with a distance quotient threshold of 1.0, averaged over unknown senses

[4]Conditions `Syn` and `Syn-hw` were also tested using only other target lemmas with the same part of speech. Results were virtually unchanged.

Features	Precision all	Precision ≥ 50	Precision ≥ 200	Recall all	Recall ≥ 50	Recall ≥ 200
All	0.77	0.77	0.73	0.72	0.80	0.87
Cx	0.77	0.77	0.73	0.82	0.89	0.94
Syn	0.86	0.85	0.82	0.17	0.16	0.13
Syn-hw	0.80	0.79	0.76	0.38	0.36	0.38
Syn	0.86	0.85	0.82	0.17	0.17	0.14
Syn-hw	0.81	0.80	0.76	0.35	0.37	0.38

Table 6: Experiment 3: Results by training set size, $\theta = 1.0$

		Number of senses 2	3	4	5
Exp. 2	Prec.	0.78	0.68	0.59	0.55
	Rec.	0.21	0.38	0.47	0.59
Exp. 3	Prec.	0.83	0.71	0.63	0.56
	Rec.	0.68	0.81	0.89	0.88

Table 7: Experiments 2 and 3: Results by the number of senses of a lemma, condition All, $\theta = 1.0$

and 5-fold cross-validation. In comparison to Experiment 2, precision has risen slightly, and for conditions All, Cx and Syn-hw, recall has risen steeply; the maximum recall is achieved by Cx at 0.82.

As before, increasing the distance quotient threshold leads to little change in precision but a sharp drop in recall. For All vectors, recall is 0.72 for threshold 1.0, 0.56 for $\theta = 1.1$, and 0.41 for $\theta = 1.2$.

Table 6 shows system performance by training set size. As the average training set in this experiment is much larger than in Experiment 2, we are now inspecting sets of minimum size 50 and 200 rather than 10 and 20. We find the same effect as in Experiment 2, with noticeably higher recall for lemmas with larger training sets, but slightly lower precision.

Table 7 breaks down system performance by the degree of ambiguity of a lemma. Here, too, we see the same effect as in Experiment 2: the more senses a lemma has, the lower the precision and the higher the recall of *unknown* label assignment.

Discussion. In comparison to Experiment 2, Experiment 3 shows a dramatic increase in recall, and even some increase in precision. Precision and recall for conditions All and Cx are good enough for the system to be usable in practice.

Of the four conditions, the three that involve context words, All, Cx and Syn-hw, show considerably higher recall than Syn. Furthermore, the two conditions that do not involve syntactic features, All and Cx, have markedly higher results than Syn-hw. This could mean that syntactic features are not as helpful as context features in detecting unknown senses; however in Experiment 2 the performance difference between Syn and the other conditions was not by far as large as in this experiment. It could also mean that frames are not as uniform in their syntactic structure as they are in their context words. This seems plausible as FrameNet frames are constructed mostly on semantic grounds, without recourse to similarity in syntactic structure.

Table 6 points to a sparse data problem, even with training sets extended by additional items. It also shows that the more a test condition relies on context word information, the more it profits from additional data. So it may be worthwhile to explore methods for a further alleviation of data sparseness, e.g. by generalizing over context words.

Table 7 underscores the large influence of training set uniformity: the more senses a lemma has, the more likely the model is to classify a test instance as *unknown*. This is the case even for extended training sets. One possible way of addressing this problem would be to take into account more than a single nearest neighbor in NN-based outlier detection in order to compute more precise boundaries between known and unknown instances.

8 Conclusion and outlook

We have defined and addressed the problem of *unknown word sense detection*: the identification of corpus occurrences that are not covered by a given sense inventory, using a training set of sense-annotated data as a basis. We have modeled this problem as an instance of *outlier detection*, using the simple nearest neighbor-based approach of Tax and Duin to measure the resemblance of a new occurrence to the training data. In combination with a method that alleviates data sparseness by sharing training data across lemmas, the approach achieves good results that make it usable in practice: With items represented as vectors of context words (including lemma, POS and NE), the system achieves 0.77 precision and 0.82 recall in an evaluation on FrameNet 1.2. The training set extension method,

which proved crucial to our approach, relies solely on a grouping of annotated data by semantic similarity. As such, the method is applicable to any resource that groups words into semantic classes, for example WordNet.

For this first study on unknown sense detection, we have chosen a maximally simple outlier detection method; many extensions are possible. One obvious possibility is the extension of Tax and Duin's method to more than one nearest training neighbor for a more accurate estimate of local density. Furthermore, more sophisticated feature vectors can be employed to generalize over context words, and other outlier detection approaches (Markou and Singh, 2003a; Markou and Singh, 2003b; Marsland, 2003) can be tested on this task.

Our immediate goal is to use unknown sense detection in combination with WSD, to filter out items that the WSD system cannot handle due to missing senses. Once items have been identified as *unknown*, they are available for further processing: If possible one would like to assign some measure of sense information even to these items. Possibilities include associating items with similar existing senses (Widdows, 2003; Curran, 2005; Burchardt et al., 2005) or clustering them into approximate senses.

References

C. Baker, C. Fillmore, and J. Lowe. 1998. The Berkeley FrameNet Project. In *Proc. ACL-98*, Montreal.

A. Burchardt, K. Erk, and A. Frank. 2005. A WordNet detour to FrameNet. In *Proc. GLDV 2005 Workshop GermaNet II*, Bonn.

U. Callmeier, A. Eisele, U. Schäfer, and M. Siegel. 2004. The DeepThought core architecture framework. In *Proc. LREC-04*, Lisbon.

James Curran. 2005. Supersense tagging of unknown nouns using semantic similarity. In *Proc. ACL-05*, Ann Arbor.

D. Dasgupta and S. Forrest. 1999. Novelty detection in time series data using ideas from immunology. In *Proc. International Conference on Intelligent Systems*.

Katrin Erk and Sebastian Pado. 2006. Shalmaneser - a toolchain for shallow semantic parsing. In *Proc. LREC-06*, Genoa.

K. Erk. 2005. Frame assignment as word sense disambiguation. In *Proc. IWCS 2005*, Tilburg.

C. Fillmore. 1982. Frame Semantics. *Linguistics in the Morning Calm*.

R. Florian, S. Cucerzan, C. Schafer, and D. Yarowsky. 2002. Combining classifiers for word sense disambiguation. *Journal of Natural Language Engineering*, 8(4):327–431.

S. Hickinbotham and J. Austin. 2000. Neural networks for novelty detection in airframe strain data. In *Proc. International Joint Conference on Neural Networks*.

D. Lin. 1993. Principle-based parsing without overgeneration. In *Proc. ACL-93*, Columbus, OH.

M. Markou and S. Singh. 2003a. Novelty detection: A review. part 1: Statistical approaches. *ACM Signal Processing*, 83(12):2481 – 2497.

M. Markou and S. Singh. 2003b. Novelty detection: A review. part 2: Neural network based approaches. *ACM Signal Processing*, 83(12):2499 – 2521.

S. Marsland. 2003. Novelty detection in learning systems. *Neural computing surveys*, 3:157–195.

D. Mount and S. Arya. 2005. ANN: A library for approximate nearest neighbor searching. Download from `http://www.cs.umd.edu/~mount/ANN/`.

B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt. 2000. Support vector method for novelty detection. *Advances in neural information processing systems*, 12.

H. Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97 – 123.

M. Tatu and D. Moldovan. 2005. A semantic approach to recognizing textual entailment. In *Proc. HLT/EMNLP 2005*, Vancouver.

D. Tax and R. Duin. 1998. Outlier detection using classifier instability. In *Advances in Pattern Recognition: the Joint IAPR International Workshops*.

D. Tax and R. Duin. 2000. Data description in subspaces. In *International Conference on Pattern recognition*, volume 2, Barcelona.

Dominic Widdows. 2003. Unsupervised methods for developing taxonomies by combining syntactic and statistical information. In *Proc. HLT/NAACL-03*, Edmonton.

D. Yeung and C. Chow. 2002. Parzen-window network intrusion detectors. In *Proc. International Conference on Pattern Recognition*.

Understanding Temporal Expressions in Emails

Benjamin Han, Donna Gates and Lori Levin
Language Technologies Institute
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh PA 15213
{benhdj|dmg|lsl}@cs.cmu.edu

Abstract

Recent years have seen increasing research on extracting and using temporal information in natural language applications. However most of the works found in the literature have focused on identifying and understanding temporal expressions in *newswire texts*. In this paper we report our work on *anchoring* temporal expressions in a novel genre, *emails*. The highly under-specified nature of these expressions fits well with our constraint-based representation of time, *Time Calculus for Natural Language* (TCNL). We have developed and evaluated a Temporal Expression Anchoror (TEA), and the result shows that it performs significantly better than the baseline, and compares favorably with some of the closely related work.

1 Introduction

With increasing demand from ever more sophisticated NLP applications, interest in extracting and understanding temporal information from texts has seen much growth in recent years. Several works have addressed the problems of representing temporal information in natural language (Setzer, 2001; Hobbs and Pan, 2004; Saurí et al., 2006), extracting and/or anchoring (normalizing) temporal and event related expressions (Wiebe et al., 1998; Mani and Wilson, 2000; Schilder and Habel, 2001; Vazov, 2001; Filatova and Hovy, 2001), and discovering the ordering of events (Mani et al., 2003). Most of these works have focused on capturing temporal information contained in *newswire texts*, and whenever both *recognition* and *normalization* tasks of temporal expressions were attempted, the latter almost always fell far behind from the former in terms of performance.

In this paper we will focus on a different combination of the problems: anchoring temporal expressions in *scheduling-related emails.* In our project work of building personal agents capable of scheduling meetings among different users[1], understanding temporal expressions is a crucial step. We have therefore developed and evaluated our system Temporal Expression Anchorer (TEA) that is capable of normalizing such expressions in texts. As input TEA takes English text with temporal expressions already identified, and transduces the expressions into their representations using *Time Calculus for Natural Language* (TCNL) (Han and Kohlhase, 2003). These representations, or TCNL formulae, are then evaluated by incorporating the contextual information to give the final normalized output. TCNL has the following characteristics: (1) a human calendar (e.g., the Gregorian calendar) is explicitly modeled as a constraint system to deal with the highly under-specified nature of many temporal expressions, and it allows easy extension to include new temporal primitives; (2) a set of NL-motivated operators with a granularity-enriched type system facilitates the representation of the *intensional* meaning of a tem-

[1] Project RADAR, http://www.radar.cs.cmu.edu/external.asp

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 136–143,
New York, June 2006.

poral expression in a compositional way; and (3) the use of temporal references such as "focus" in the representation cleanly separates the core meaning of an expression from its contextual dependency.

The rest of this paper is organized as follows. Sec. 2 first surveys the characteristics of temporal expressions in emails compared to those in newswire texts, and motivates the design of our representation. Sec 3 then introduces the formalism TCNL. The system TEA and the anchoring process is detailed in Sec. 4, and the evaluation of the system is reported in Sec. 5. Finally Sec. 6 concludes this paper and outlines the future work.

2 Temporal Expressions in Emails

The extent of temporal expressions considered in this paper includes most of the expressions using temporal terms such as *2005*, *summer*, *evening*, *1:30pm*, *tomorrow*, etc. These expressions can be classified into the following categories:

- **Explicit**: These expressions can be immediately anchored, i.e., positioned on a timeline. E.g., *June 2005*, *1998 Summer*, etc.
- **Deictic**: These expressions form a specific relation with the *speech time* (timestamp of an email). E.g., *tomorrow*, *last year*, *two weeks from today*.
- **Relative**: These include the other expressions that form a specific relation with a *temporal focus*, i.e., the implicit time central to the discussion. E.g., *from 5 to 7*, *on Wednesday*, etc. Different from the speech time, a temporal focus can shift freely during the discourse.
- **Durational**: These are the expressions that describe certain length in time. E.g., *for about an hour*, *less than 20 minutes*. This is different from an interval expression where both the starting point and the ending point are given (e.g., *from 5 to 7*). Most durational expressions are used to build more complex expressions, e.g., *for the next 20-30 minutes.*

It is worth emphasizing the crucial difference between deictic expressions and relative expressions: anchoring the former only relies on the fixed speech time while normalizing the latter requires the usually hidden focus. As illustrated below the latter task can be much more challenging:

> "*I'm free next week. Let's meet* ***on Wednesday***."
> "*Are you free* ***on Wednesday****?*"

In the first example the "*Wednesday*" denotes a different date since the first sentence sets up a different focus. To make things even more interesting, verbal tense can also play a role, e.g., "*He finished the report* ***on Wednesday***."

There are other types of temporal expressions such as *recurrence* ("*every Tuesday*") and *rate* expressions ("*twice on Wednesday*") that are not supported in our system, although they are planned in our future work (Sec. 6).

To appreciate the different nature of emails as a genre, an interesting observation can be made by comparing the distributions of temporal expressions in emails and in *newswire texts*. The email corpora we used for development and testing were collected from MBA students of Carnegie Mellon University over the year 1997 and 1998. The 277 students, organized in approximately 50 teams of 4 to 6 members, were participating in a 14-week course and running simulated companies in a variety of market scenarios (Kraut et al., 2004). The original dataset, the CSpace email corpus, contains approximately 15,000 emails. We manually picked 1,196 emails that are related to scheduling - these include scheduling meetings, presentations, or general planning for the groups. The emails are then randomly divided into five sets (**email1** to **email5**), and only four of them are used in this work: **email1** was used to establish our baseline, **email2** and **email5** were used for development, and part of **email4** was used for testing. Table 1 shows some basic statistics of these three datasets[2], and an edited sample email is shown in Fig. 1 (names altered). The most apparent difference comparing these emails to newswire texts is in the percentage of explicit expressions occurring in the two different genres. In (Mani et al., 2003) it was reported that the proportion of such expressions is about 25% in the newswire corpus they

[2]The percentages in some rows do not add up to 100% because some expressions like coordination can be classified into more than one type.

Date: **Thu, 11 Sep 1997 00:14:36 -0500**

I have put an outline out in the n10f1 OpReview directory... *(omitted)*

We have very little time for this. Please call me **Thursday night** to get clarification. I will need graphs and prose in files **by Saturday Noon.**

– Mary

ps. Mark and John , I waited **until AFTER midnight** to send this .

Figure 1: A sample email (edited)

used[3]. In contrast, explicit expressions on average only account for around 9.5% in the three email datasets. This is not surprising given that people tend to use under-specified expressions in emails for economic reasons. Another thing to note is that there are roughly the same number of relative expressions and non-relative expressions. Since non-relative expressions (including deictic expressions) can be anchored without tracking the temporal focus over a discourse and therefore can be dealt with in a fairly straightforward way, we may assign 50% as a somewhat generous baseline performance of any anchoring system[4].

Another difference between emails and newswire texts is that the former is a medium for *communication*: an email can be used as a reply, or can be attached within another email, or even be used to address to multiple recipients. All of this complicates a great deal of our task. Other notable differences are that in emails hour ambiguity tend to appear more often ("*I'll be home at 2*."), and people tend to be more creative when they compose short messages such as using tables (e.g., an entire column of numbers to denote the number of minutes alloted for each presenter), bullet lists, abbreviations, and different month/day formats ("*1/9*" can mean *January 9* or *September 1*), etc. Emails also contain more "human errors" such as misspellings ("*Thusday*" to mean *Thursday*) and confusion about dates (e.g., using "*tomorrow*" when sending emails around midnight), etc. Overall it is very difficult to recover from this type of errors.

3 Representing Times in Natural Language

This section provides a concise overview of TCNL; readers are referred to (Han and Kohlhase, 2003; Han et al., 2006) for more detail.

TCNL has two major components: a constraint-based model for human calendars and a representational language built on top of the model. Different from the other representations such as TeitGram (Stede and Haas, 1998), TOP (Androutsopoulos, 1999), and TimeML/Timex3 (Saurí et al., 2006), the language component of TCNL is essentially "calendar-agnostic" - any *temporal unit* can be plugged in a formula once it is defined in the calendar model, i.e., the calendar model serves as the lexicon for the TCNL language.

Fig. 2 shows a partial model for the Gregorian calendar used in TEA. The entire calendar model is basically a constraint graph with *partial ordering*. The nodes labeled with "`year`" etc. represent temporal units (or variables when viewed as a constraint satisfaction problem (CSP) (Ruttkay, 1998)), and each unit can take on a set of possible values. The undirected edges represent constraints among the units, e.g., the constraint between `month` and `day` mandates that February cannot have more than 29 days. A temporal expression in NL is then viewed as if it assigns values to some of the units, e.g., "*Friday the 13th*" assigns values to only units `dow` (day-of-week) and `day`. An interval-based AC-3 algorithm with a chronological backtracking mechanism is used to derive at the consistent assignments to the other units, therefore allowing us to iterate to any one of the possible *Friday the 13th*.

The ordering among the units is designated by two relations: *measurement* and *periodicity* (arrows in Fig. 2). These relations are essential for supporting various operations provided by the TCNL language such as determining temporal ordering of two time points, performing arithmetic, and changing temporal granularity, etc. For example, to interpret the expression "*early July*", we identify that *July* is a value of unit `month`, and `month` is measured by `day`. We then obtain the size of *July* in terms of `day` (31) and

[3]Using the North American News Corpus.

[4]This is a bit generous since solving simple calendric arithmetics such as anchoring *last summer* still requires a non-trivial modeling of human calendars; see Sec. 3.

Table 1: Basic statistics of the email corpora

	# of emails	# of tempex	explicit	deictic	relative	durational
email1	253	300	3 (1%)	139 (46.33%)	158 (52.67%)	N/A
email2	253	344	19 (5.5%)	112 (32.6%)	187 (54.4%)	27 (7.8%)
email4 (part.)	149	279	71 (25.4%)	77 (27.6%)	108 (38.7%)	22 (7.9%)
email5	126	213	14 (6.6%)	105 (49.3%)	92 (43.2%)	3 (1.4%)

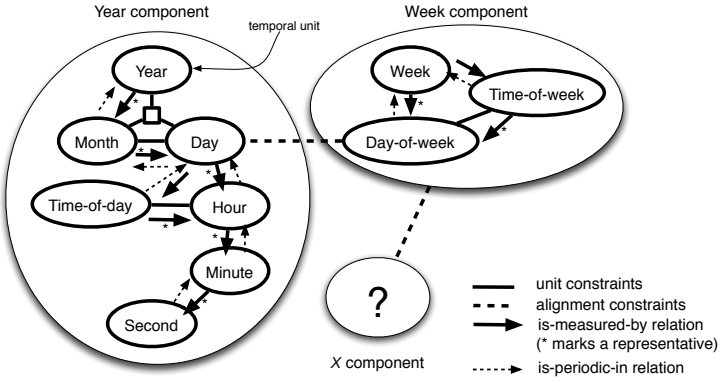


Figure 2: A partial model of the Gregorian calendar

designate the first 10 days (31/3) as the "early" part of July.

Internally the calendar model is further partitioned into several *components*, and different components are *aligned* using non-binary constraints (e.g., in Fig. 2 the `year` component and the `week` component are aligned at the `day` and `dow` units). This is necessary because the top units in these component are not periodic within one another. All of the operations are then extended to deal with multiple calendar components.

Built on top of the calendar model is the *typed* TCNL language. The three major types are *coordinates* (time points; e.g., $\{\texttt{sep}, 6_{\texttt{day}}\}$ for *September 6*), *quantities* (durations; e.g., $|1_{\texttt{hour}}|$ for *one hour*) and *enumerations* (sets of points, including intervals; e.g., $[\{\texttt{wed}\},\{\texttt{fri}\}]$ for *Wednesday and Friday*). More complex expressions can be represented by using various operators, relations and temporal references; e.g., $\{\texttt{now}-|1_{\texttt{day}}|\}$ for *yesterday*, $\{|1_{\texttt{mon}}|@\{>= \_\}\}$ for *the coming Monday* (or *the first coming Monday in the future*; the '_' represents the temporal focus), $|<|1_{\texttt{hour}}||$ for *less than one hour*, $[\{\texttt{wed}\}:\{\texttt{fri}\}]$ for *Wednesday to Friday*, $[\texttt{f}\ \{\texttt{sat}, \texttt{noon}\}]$ for *by Saturday noon*[5], and $[[\{15_{\texttt{hour}}\}:\{17_{\texttt{hour}}\}]\&\{\texttt{wed}\}]$ for *3-5pm on Wednesday.* The TCNL language is designed in such a way that syntactically different formulae can be evaluated to denote the same date; e.g., $\{\texttt{tue}, \texttt{now}+|1_{\texttt{week}}|\}$ ("*Tuesday next week*") and $\{\texttt{now}+|1_{\texttt{tue}}|\}$ ("*next Tuesday*") can denote the same date.

Associated with the operators are type and granularity requirements. For example, when a focus is specified down to *second* granularity, the formula $\{\texttt{now}+|1_{\texttt{day}}|\}$ will return a coordinate at the *day* granularity - essentially stripping away information finer than *day*. This is because the operator '+' (called *fuzzy forward shifting*) requires the left-hand side operand to have the same granularity as that of the right-hand side operand. Type coercion can also happen automatically if it is required by an operator. For example, the operator '@' (*ordinal selection*) requires that the right-hand side operand to be of type enumeration. When presenting a coordinate such as $\{>= \_\}$ (some point in the future), it will be coerced

[5]The `f` denotes the relation "*finishes*" (Allen, 1984); the formula denotes a set of coordinates no later than a Saturday noon.

Table 2: Summary of operators in TCNL; LHS/RHS is the left/right operand, g(e) returns the granularity of e and $\min(s)$ returns the set of minimal units among s.

operator	Type requirement	Granularity requirement	Semantics	Example
$+$ and $-$	C $\times$ Q $\rightarrow$ C	g(LHS) $\leftarrow$ g(RHS)	fuzzy forward/backward shifting	{now+|$1_{\texttt{day}}$|} ("*tomorrow*")
$++$ and $--$	C $\times$ Q $\rightarrow$ C	g(LHS) $\leftarrow$ $\min$(g(LHS)$\cup$g(RHS))	exact forward/backward shifting	{now++|$2_{\texttt{hour}}$|} ("*2 hours from now*")
@	Q $\times$ E $\rightarrow$ C	g(RHS) $\leftarrow$ g(LHS)	ordinal	{|$2_{\{\texttt{sun}\}}$|@{may}} ("*the 2nd Sunday in May*")
&	C $\times$ C $\rightarrow$ C C $\times$ E $\rightarrow$ E E $\times$ C $\rightarrow$ E E $\times$ E $\rightarrow$ E	g(LHS) $\leftarrow$ $\min$(g(LHS)$\cup$g(RHS))	distribution	{now &{now+|$1_{\texttt{year}}$|}} ("*this time next year*") [{$15_{\texttt{hour}}$}&[{wed}:{fri}]] ("*3pm from Wednesday to Friday*")

into an enumeration so that the ordinal operator can select a requested element out of it. These designs make granularity change and re-interpretation part of a transparent process. Table 2 lists the operators in the TCNL language.

Most of under-specified temporal expressions still lack necessary information in themselves in order to be anchored. For example, it is not clear what to make out of "*on Wednesday*" with no context. In TCNL more information can be supplied by using one of the coordinate *prefixes*: the '+'/'−' prefix signifies the relation of a coordinate with the focus (after/before the focus), and the 'f'/'p' indicates the relation of a coordinate with the speech time (future/past). For example, the Wednesday in "*the company will announce on Wednesday*" is represented as +f{wed}, while "*the company announced on Wednesday*" is represented as −p{wed}. When evaluating these formulae, TEA will rewrite the former into {\|$1_{\texttt{wed}}$\|@{>= _, >= now}} and the latter into {−\|$1_{\texttt{wed}}$\|@{<= _, <= now}} if necessary, essentially trying to find the nearest Wednesday either in the future or in the past. Since TCNL formulae can be embedded, prefixed coordinates can also appear inside a more complex formula; e.g., {{\|$2_{\{\texttt{sun}\}}$\|@f{may}}+\|$2_{\texttt{day}}$\|} represents "*2 days after a future Mother's day*"[6].

Note that TCNL itself does not provide a mechanism to instantiate the temporal focus ('_'). The responsibility of shifting a focus whenever necessary (*focus tracking*) is up to TEA, which is described in the next section.

[6]This denotes a possible range of dates, but it is still different from an enumeration.

4 TEA: Temporal Expression Anchorer

The input to our system TEA is English texts with temporal expression markups, and the output is a time string for each temporal expression. The format of a time string is similar to the ISO 8601 scheme: for a time point the format is YYYYMMDDTHHMMSS (T is a separator), for an interval it is a pair of points separated by '/' (slash). Also whenever there are slots that lack information, we use '?' (question mark) in its place. If a points can reside at any place between two bounds, we use (lower..upper) to represent it. Table. 3 shows the TEA output over the example email given in Fig. 1 (min and max are the minimal and the maximal time points TEA can reason with).

TEA uses the following procedure to anchor each temporal expression:

1. The speech time (variable now) and the focus ('_') is first assigned to a timestamp (e.g., the received date of an email).

2. For each temporal expression, its nearest verb chunk is identified using the part-of-speech tags of the sentence. Expressions associated with a verb of past tense or present imperfective will be given prefix "−p" to its TCNL formula, otherwise it is given "+f"[7].

3. A finite-state parser is then used to transduce an expression into its TCNL formula. At the parsing stage the tense and granularity information is available to the parser.

[7]This is of course a simplification; future work needs to be done to explore other possibilities.

Table 3: Anchoring example for the email in Fig. 1

Expression	TCNL formula	Temporal focus (f)	Anchored time string
(timestamp)			19970911T001436
Thursday night	$+\texttt{f}\{\texttt{thu,night}\}$	19970911T001436	(19970911T18????.. 19970911T23????)
by Saturday Noon	$[\texttt{f} + \texttt{f}\{\texttt{sat,noon}\}]$	(19970911T18????.. 19970911T23????)	min/19970913T12????
until AFTER midnight	$[\texttt{f}\{>= -\texttt{p}\{\texttt{midnight}\}\}]$	19970911T001436	min/(19970911..max)

4. The produced TCNL formula (or formulae when ambiguity arises) is then evaluated with the speech time and the current focus. In case of ambiguity, one formula will be chosen based on certain heuristics (below). The result of the evaluation is the final output for the expression.

5. *Recency-based* focus tracking: we use the following procedure to determine if the result obtained above can replace the current focus (below). In cases where the result is an ambiguous coordinate (i.e., it denotes a possible range of points), if one of the bounds is `min` or `max`, we use the other to be the new focus; if it is not possible, we choose to keep the focus unchanged. On the other hand, if the result is an enumeration, we go through a similar procedure to avoid using an enumeration with a `min`/`max` bound as the new focus. Finally no quantity can become a focus.

Note that in Step 3 the decision to make partial semantics of a temporal expression available to our parser is based on the following observation: consider the two expressions below

"*Tuesday before Christmas*"
$= \{\texttt{tue}, < \{|25_{\texttt{day}}|@\{\texttt{dec}\}\}\}$

"*Tuesday before 6pm*"
$= \{< \{\texttt{tue},18_{\texttt{hour}}\}, \textbf{de}\ \{\texttt{tue}\}\}$

Both expressions share the same "X before Y" pattern, but their interpretations are different[8]. The key to discriminate the two is to compare the granularities of X and Y: if Y if at a coarser granularity then the first interpretation should be adopted.

In Step 4 we use the following procedure to disambiguate the result:

1. Remove any candidate that resulted in an inconsistency when solving for a solution in the calendar CSP.

2. If the result is meant to be a coordinate, pick the one that is closest to the focus.

3. If the result is supposed to be an enumeration, pick the one whose starting point is closest to the focus, and whose length is the shortest one.

4. Otherwise pick the first one as the result.

For example, if the current time is 2:00 pm, for expression "*at 3*" with a present/future tense, the best answer is 15:00. For expression "*from 3 to 5*", the best answer is from 3 pm to 5 pm.

When deciding whether a temporal expression can become the next focus, we use simple heuristics to rule out any expression that behaves like a noun modifier. This is motivated by the following example (timestamp: `19970919T103315`):

> *IT basically analyses the breakdown on labor costs and compares our **1998** labor costs with their demands for **1999-2000**.*
> ...
> *I will check mail **on Sunday** and see any feedback.*

Without blocking the expression *1999-2000* from becoming the focus, the last expression will be incorrectly anchored in year 2000. The key observation here is that a noun-modifying temporal expression usually serves as a temporal co-reference instead of representing a new temporal entity in the discourse. These references tend to have a more confined effect in anchoring the subsequent expressions.

[8] **de** denotes a relation "during or equal" (Allen, 1984).

Table 4: Development and testing results

	Accuracy	Parsing errors	Human errors	Anchoring errors
email2 (dev)	78.2%	10.47%	1.7%	9.63%
email5 (dev)	85.45%	5.16%	1%	8.39%
email4 (testing)	76.34%	17.92%	$< 1\%$	5.74%

5 Evaluation

The temporal expressions in all of the datasets were initially tagged using rules developed for MinorThird[9], and subsequently corrected manually by two of the authors. We then developed a prototype system and established our baseline over **email1** (50%). The system at that time did not have any focus tracking mechanism (i.e., it always used the timestamp as the focus), and it did not use any tense information. The result confirms our estimate given in Sec. 2. We then gradually developed TEA to its current form using **email1**, **email2** and **email5**. During the four-month development we added the focus tracking mechanism, incorporating the tense information into each TCNL formula via the coordinate prefixes, and introduced several representational improvements. Finally we tested the system on the unseen dataset **email4**, and obtained the results shown in Table 4. Note that the percentages reported in the table are *accuracies*, i.e., the number of correctly anchored expressions over the total number of temporal expressions over a dataset, since we are assuming correct tagging of all of the expressions. Our best result was achieved in the dev set **email5** (85.45%), and the accuracy over the test set **email4** was 76.34%.

Table 4 also lists the types of the errors made by our system. The parsing errors are mistakes made at transducing temporal expressions using the finite-state parser into their TCNL formulae, the human errors are described in Sec. 2, and the rest are the anchoring errors. The accuracy numbers are all compared favorably to the baseline (50%). To put this performance in perspective, in (Wiebe et al., 1998) a similar task was performed over transcribed scheduling-related phone conversations. They reported an average accuracy 80.9% over the CMU test set and 68.9% over the NMSU test set. Although strictly speaking the two results cannot be compared due to differences in the nature of the corpora (transcription vs. typing), we nevertheless believe it represents a closer match compared to the other works done on newswire genre.

It should also be noted that we adopted a similar recency-based focus model as in (Wiebe et al., 1998). Although simple to implement, this naive approach proved to be one major contributor to the anchoring errors in our experiments. An example is given below (the anchored times are shown in subscript):

> *This research can not proceed until the trade-offs are known* ***on Monday***$_{19970818}$.
> ...
> *Mary will perform this* ***by Friday***$_{(min..19970822)}$ *using the data* ***from Monday***$_{19970825}$.

The last expression received an incorrect date: it should be the same date the expression "*on Monday*" refers to. Our system made this error because it blindly used the most recently mentioned time (`((min..19970822))`) as the focus to anchor the formula $+\mathtt{f}\{\mathtt{mon}\}$. This error later also propagated to the anchoring of the subsequent expressions.

6 Conclusion and Future Work

In this paper we have adopted a constraint-based representation of time, *Time Calculus for Natural Language* (TCNL), to tackle the task of anchoring temporal expressions in a novel genre, *emails*. We believe that the genre is sufficiently different from newswire texts, and its highly under-specified nature fits well with a constraint-based modeling of human calendars. TCNL also allows for an explicit representation of *temporal focus*, and many of our intuitions about *granularity change* and *temporal arithe-*

[9] `http://minorthird.sourceforge.net/`

matics are encapsulated in its type system and operators. The performance of our anchoring system is significantly better than baseline, and compares favorably with some of the closely related work.

In the future we will re-examine our focus tracking mechanism (being the most significant source of errors), and possibly treat it as a classification problem (similar to (Mani et al., 2003)). We also need to investigate the disambiguation procedure and possibly migrate the functionality into a separate discourse module. In addition, the co-referencing tendency of noun-modifying expressions could lead to a better way to anchoring this particular type of temporal expressions. Finally we would like to expand our coverage of temporal expressions to include other types of expressions such as recurrence expressions[10].

Acknowledgments

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA), or the Department of Interior-National Business Center (DOI-NBC).

References

J. F. Allen. 1984. Towards a General Theory of Action and Time. *Artificial Intelligence*, 23:123–154.

I. Androutsopoulos. 1999. Temporal Meaning Representations in a Natural Language Front-end. In M. Gergatsoulis and P. Rondogiannis, editors, *Intensional Programming II (Proceedings of the 12th International Symposium on Languages for Intensional Programming*, Athens, Greece.

E. Filatova and E. Hovy. 2001. Assigning Time-Stamps To Event-Clauses. In *Proceedings of ACL-2001: Workshop on Temporal and Spatial Information Processing*, Toulouse, France, 7.

Benjamin Han and Michael Kohlhase. 2003. A Time Calculus for Natural Language. In *The 4th Workshop on Inference in Computational Semantics*, Nancy, France, September.

B. Han, D. Gates, and L. Levin. 2006. From Language to Time: A Temporal Expression Anchorer. In *Proceedings of the 13th International Symposium on Temporal Representation and Reasoning (TIME 2006)*, Budapest, Hungary.

J. R. Hobbs and Feng. Pan. 2004. An ontology of time for the semantic web. *TALIP Special Issue on Spatial and Temporal Information Processing*, 3(1):66–85, March.

R. E. Kraut, S. R. Fussell, F. J. Lerch, and A Espinosa. 2004. Coordination in teams: Evidence from a simulated management game. *Journal of Organizational Behavior*, to appear.

I. Mani and G. Wilson. 2000. Robust Temporal Processing of News. In *Proceedings of ACL-2000*.

I. Mani, B. Schiffman, and J. Zhang. 2003. Inferring Temporal Ordering of Events in News. In *Proceedings of the Human Language Technology Conference (HLT-NAACL'03)*.

Zsófia Ruttkay. 1998. Constraint Satisfaction - a Survey. Technical Report 11(2-3), CWI.

Roser Saurí, Jessica Littman, Bob Knippen, Robert Gaizauskas, Andrea Setzer, and James Pustejovsky, 2006. *TimeML Annotation Guidelines, Version 1.2.1*, January 31.

F. Schilder and C. Habel. 2001. From Temporal Expressions To Temporal Information: Semantic Tagging Of News Messages. In *Proceedings of ACL-2001: Workshop on Temporal and Spatial Information Processing*, Toulouse, France, 7.

Andrea Setzer. 2001. *Temporal Information in Newswire Articles: an Annotation Scheme and Corpus Study*. Ph.D. thesis, University of Sheffield.

M. Stede and S. Haas. 1998. Understanding and tracking temporal descriptions in dialogue. In B. Schröder, W. Lenders, W. Hess, and T. Portele, editors, *Proceedings of the 4th Conference on Natural Language Processing - KONVENS '98*.

N. Vazov. 2001. A System for Extraction of Temporal Expressions from French Texts Based on Syntactic and Semantic Constraints. In *Proceedings of ACL-2001: Workshop on Temporal and Spatial Information Processing*, Toulouse, France, 7.

J. M. Wiebe, T. P. O'Hara, T. Ohrstrom-Sandgren, and K. J. McKeever. 1998. An Empirical Approach to Temporal Reference Resolution. *Journal of Artificial Intelligence Research*, 9:247–293.

[10]The current design of TCNL allows for a more restricted type of recurrence: e.g., "*3pm from Wednesday to Friday*" is represented as [{15$_{\texttt{hour}}$}&[{wed}:{fri}]]. However this is insufficient to represent expressions such as "*every 4 years*".

Partial Training for a Lexicalized-Grammar Parser

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

We propose a solution to the annotation bottleneck for statistical parsing, by exploiting the lexicalized nature of Combinatory Categorial Grammar (CCG). The parsing model uses predicate-argument dependencies for training, which are derived from sequences of CCG lexical categories rather than full derivations. A simple method is used for extracting dependencies from lexical category sequences, resulting in high precision, yet incomplete and noisy data. The dependency parsing model of Clark and Curran (2004b) is extended to exploit this partial training data. Remarkably, the accuracy of the parser trained on data derived from category sequences alone is only 1.3% worse in terms of F-score than the parser trained on complete dependency structures.

1 Introduction

State-of-the-art statistical parsers require large amounts of hand-annotated training data, and are typically based on the Penn Treebank, the largest treebank available for English. Even robust parsers using linguistically sophisticated formalisms, such as TAG (Chiang, 2000), CCG (Clark and Curran, 2004b; Hockenmaier, 2003), HPSG (Miyao et al., 2004) and LFG (Riezler et al., 2002; Cahill et al., 2004), often use training data derived from the Penn Treebank. The labour-intensive nature of the treebank development process, which can take many years, creates a significant barrier for the development of parsers for new domains and languages.

Previous work has attempted parser adaptation without relying on treebank data from the new domain (Steedman et al., 2003; Lease and Charniak, 2005). In this paper we propose the use of annotated data in the new domain, but only *partially annotated* data, which reduces the annotation effort required (Hwa, 1999). We develop a parsing model which can be trained using partial data, by exploiting the properties of lexicalized grammar formalisms. The formalism we use is Combinatory Categorial Grammar (Steedman, 2000), together with a parsing model described in Clark and Curran (2004b) which we adapt for use with partial data.

Parsing with Combinatory Categorial Grammar (CCG) takes place in two stages: first, CCG *lexical categories* are assigned to the words in the sentence, and then the categories are combined by the parser (Clark and Curran, 2004a). The lexical categories can be thought of as detailed part of speech tags and typically express subcategorization information. We exploit the fact that CCG lexical categories contain a lot of syntactic information, and can therefore be used for training a full parser, even though attachment information is not explicitly represented in a category sequence. Our partial training regime only requires sentences to be annotated with lexical categories, rather than full parse trees; therefore the data can be produced much more quickly for a new domain or language (Clark et al., 2004).

The partial training method uses the log-linear dependency model described in Clark and Curran (2004b), which uses sets of predicate-argument de-

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 144–151,
New York, June 2006.

pendencies, rather than derivations, for training. Our novel idea is that, since there is so much information in the lexical category sequence, most of the correct dependencies can be easily inferred from the categories alone. More specifically, for a given sentence and lexical category sequence, we train on those predicate-argument dependencies *which occur in k% of the derivations licenced by the lexical categories*. By setting the k parameter high, we can produce a set of high precision dependencies for training. A similar idea is proposed by Carroll and Briscoe (2002) for producing high precision data for lexical acquisition.

Using this procedure we are able to produce dependency data with over 99% precision and, remarkably, up to 86% recall, when compared against the complete gold-standard dependency data. The high recall figure results from the significant amount of syntactic information in the lexical categories, which reduces the ambiguity in the possible dependency structures. Since the recall is not 100%, we require a log-linear training method which works with partial data. Riezler et al. (2002) describe a partial training method for a log-linear LFG parsing model in which the "correct" LFG derivations for a sentence are those consistent with the less detailed gold standard derivation from the Penn Treebank. We use a similar method here by treating a CCG derivation as correct if it is *consistent with* the high-precision partial dependency structure. Section 3 explains what we mean by consistency in this context.

Surprisingly, the accuracy of the parser trained on partial data approaches that of the parser trained on full data: our best partial-data model is only 1.3% worse in terms of dependency F-score than the full-data model, despite the fact that the partial data does not contain *any* explicit attachment information.

2 The CCG Parsing Model

Clark and Curran (2004b) describes two log-linear parsing models for CCG: a normal-form derivation model and a dependency model. In this paper we use the dependency model, which requires sets of predicate-argument dependencies for training.[1]

[1]Hockenmaier and Steedman (2002) describe a generative model of normal-form derivations; one possibility for training this model on partial data, which has not been explored, is to use the EM algorithm (Pereira and Schabes, 1992).

The predicate-argument dependencies are represented as 5-tuples: $\langle h_f, f, s, h_a, l \rangle$, 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 non-local. For example, the dependency encoding *company* as the object of *bought* (as in ***IBM bought the company***) is represented as follows:

$$\langle bought_2,\ (S\backslash NP_1)/NP_2,\ 2,\ company_4,\ - \rangle \quad (1)$$

CCG dependency structures are sets of predicate-argument dependencies. We define the probability of a dependency structure as the sum of the probabilities of all those derivations leading to that structure (Clark and Curran, 2004b). "Spurious ambiguity" in CCG means that there can be more than one derivation leading to any one dependency structure. Thus, the probability of a dependency structure, π, given a sentence, S, is defined as follows:

$$P(\pi|S) = \sum_{d \in \Delta(\pi)} P(d, \pi|S) \quad (2)$$

where $\Delta(\pi)$ is the set of derivations which lead to π.

The probability of a $\langle d, \pi \rangle$ pair, ω, conditional on a sentence S, is defined using a log-linear form:

$$P(\omega|S) = \frac{1}{Z_S} e^{\boldsymbol{\lambda}.\boldsymbol{f}(\omega)} \quad (3)$$

where $\boldsymbol{\lambda}.\boldsymbol{f}(\omega) = \sum_i \lambda_i f_i(\omega)$. The function f_i is the integer-valued frequency function of the ith feature; λ_i is the weight of the ith feature; and Z_S is a normalising constant.

Clark and Curran (2004b) describes the training procedure for the dependency model, which uses a discriminative estimation method by maximising the conditional likelihood of the model given the data (Riezler et al., 2002). The optimisation of the objective function is performed using the limited-memory BFGS numerical optimisation algorithm (Nocedal and Wright, 1999; Malouf, 2002), which requires calculation of the objective function and the gradient of the objective function at each iteration.

The objective function is defined below, where $L(\Lambda)$ is the likelihood and $G(\Lambda)$ is a Gaussian prior term for smoothing.

He	anticipates	growth	for	the	auto	maker
$\overline{NP}$	$\overline{(S[dcl]\backslash NP)/NP}$	$\overline{NP}$	$\overline{(NP\backslash NP)/NP}$	$\overline{NP[nb]/N}$	$\overline{N/N}$	$\overline{N}$

Figure 1: Example sentence with CCG lexical categories

$$
\begin{aligned}
L'(\Lambda) &= L(\Lambda) - G(\Lambda) \qquad (4)\\
&= \sum_{j=1}^{m} \log \sum_{d \in \Delta(\pi_j)} e^{\boldsymbol{\lambda}.\boldsymbol{f}(d,\pi_j)} \\
&\quad - \sum_{j=1}^{m} \log \sum_{\omega \in \rho(S_j)} e^{\boldsymbol{\lambda}.\boldsymbol{f}(\omega)} - \sum_{i=1}^{n} \frac{\lambda_i^2}{2\sigma^2}
\end{aligned}
$$

$S_1, \ldots, S_m$ are the sentences in the training data; $\pi_1, \ldots, \pi_m$ are the corresponding gold-standard dependency structures; $\rho(S)$ is the set of possible $\langle$derivation, dependency-structure$\rangle$ pairs for S; σ is a smoothing parameter; and n is the number of features. The components of the gradient vector are:

$$
\begin{aligned}
\frac{\partial L'(\Lambda)}{\partial \lambda_i} &= \sum_{j=1}^{m} \sum_{d \in \Delta(\pi_j)} \frac{e^{\boldsymbol{\lambda}.\boldsymbol{f}(d,\pi_j)} f_i(d,\pi_j)}{\sum_{d \in \Delta(\pi_j)} e^{\boldsymbol{\lambda}.\boldsymbol{f}(d,\pi_j)}} \qquad (5)\\
&\quad - \sum_{j=1}^{m} \sum_{\omega \in \rho(S_j)} \frac{e^{\boldsymbol{\lambda}.\boldsymbol{f}(\omega)} f_i(\omega)}{\sum_{\omega \in \rho(S_j)} e^{\boldsymbol{\lambda}.\boldsymbol{f}(\omega)}} - \frac{\lambda_i}{\sigma^2}
\end{aligned}
$$

The first two terms of the gradient are expectations of feature f_i: the first expectation is over all derivations leading to each gold-standard dependency structure, and the second is over all derivations for each sentence in the training data. The estimation process attempts to make the expectations in (5) equal (ignoring the Gaussian prior term). Another way to think of the estimation process is that it attempts to put as much mass as possible on the derivations leading to the gold-standard structures (Riezler et al., 2002).

Calculation of the feature expectations requires summing over all derivations for a sentence, and summing over all derivations leading to a gold-standard dependency structure. Clark and Curran (2003) shows how the sum over the complete derivation space can be performed efficiently using a packed chart and the inside-outside algorithm, and Clark and Curran (2004b) extends this method to sum over all derivations leading to a gold-standard dependency structure.

3 Partial Training

The partial data we use for training the dependency model is derived from CCG lexical category sequences only. Figure 1 gives an example sentence adapted from CCGbank (Hockenmaier, 2003) together with its lexical category sequence. Note that, although the attachment of the prepositional phrase to the noun phrase is not explicitly represented, it can be inferred in this example because the lexical category assigned to the preposition has to combine with a noun phrase to the left, and in this example there is only one possibility. One of the key insights in this paper is that the significant amount of syntactic information in CCG lexical categories allows us to infer attachment information in many cases.

The procedure we use for extracting dependencies from a sequence of lexical categories is to return all those dependencies which occur in k% of the derivations licenced by the categories. By giving the k parameter a high value, we can extract sets of dependencies with very high precision; in fact, assuming that the correct lexical category sequence licences the correct derivation, setting k to 100 must result in 100% precision, since any dependency which occurs in every derivation must occur in the correct derivation. Of course the recall is not guaranteed to be high; decreasing k has the effect of increasing recall, but at the cost of decreasing precision.

The training method described in Section 2 can be adapted to use the (potentially incomplete) sets of dependencies returned by our extraction procedure. In Section 2 a derivation was considered correct if it produced the complete set of gold-standard dependencies. In our partial-data version a derivation is considered correct if it produces dependencies which are *consistent with* the dependencies returned by our extraction procedure. We define consistency as follows: a set of dependencies D is consistent with a set G if G is a subset of D. We also say that a derivation d is consistent with dependency set G if G is a subset of the dependencies produced by d.

This definition of "correct derivation" will introduce some noise into the training data. Noise arises from sentences where the recall of the extracted dependencies is less than 100%, since some of the derivations which are consistent with the extracted dependencies for such sentences will be incorrect. Noise also arises from sentences where the precision of the extracted dependencies is less than 100%, since for these sentences every derivation which is consistent with the extracted dependencies will be incorrect. The hope is that, if an incorrect derivation produces mostly correct dependencies, then it can still be useful for training. Section 4 shows how the precision and recall of the extracted dependencies varies with k and how this affects parsing accuracy.

The definitions of the objective function (4) and the gradient (5) for training remain the same in the partial-data case; the only differences are that $\Delta(\pi)$ is now defined to be those derivations which are consistent with the partial dependency structure π, and the gold-standard dependency structures π_j are the partial structures extracted from the gold-standard lexical category sequences.[2]

Clark and Curran (2004b) gives an algorithm for finding all derivations in a packed chart which produce a particular set of dependencies. This algorithm is required for calculating the value of the objective function (4) and the first feature expectation in (5). We adapt this algorithm for finding all derivations which are consistent with a partial dependency structure. The new algorithm is shown in Figure 2.

The algorithm relies on the definition of a packed chart, which is an instance of a *feature forest* (Miyao and Tsujii, 2002). The idea behind a packed chart is that equivalent chart entries of the same type and in the same cell are grouped together, and back pointers to the daughters indicate how an individual entry was created. Equivalent entries form the same structures in any subsequent parsing.

A feature forest is defined in terms of *disjunctive* and *conjunctive* nodes. For a packed chart, the individual entries in a cell are conjunctive nodes, and the equivalence classes of entries are disjunctive nodes. The definition of a feature forest is as follows:
A *feature forest* Φ is a tuple $\langle C, D, R, \gamma, \delta \rangle$ where:

$\langle C, D, R, \gamma, \delta \rangle$ is a packed chart / feature forest
G is a set of dependencies returned by the extraction procedure
Let c be a conjunctive node
Let d be a disjunctive node
deps(c) is the set of dependencies on node c

$$\textit{cdeps}(c) = |\textit{deps}(c) \cap G|$$

$$\textit{dmax}(c) = \textstyle\sum_{d \in \delta(c)} \textit{dmax}(d) + \textit{cdeps}(c)$$

$$\textit{dmax}(d) = \max\{\textit{dmax}(c) \mid c \in \gamma(d)\}$$

mark(d):
  mark d as a correct node
  foreach $c \in \gamma(d)$
    if *dmax*$(c) ==$ *dmax*(d)
      mark c as a correct node
      foreach $d' \in \delta(c)$
        mark(d')

foreach $d_r \in R$ such that *dmax*$(d_r) = |G|$
  mark(d_r)

Figure 2: Finding nodes in derivations consistent with a partial dependency structure

- C is a set of conjunctive nodes;
- D is a set of disjunctive nodes;
- $R \subseteq D$ is a set of root disjunctive nodes;
- $\gamma : D \rightarrow 2^C$ is a conjunctive daughter function;
- $\delta : C \rightarrow 2^D$ is a disjunctive daughter function.

Dependencies are associated with conjunctive nodes in the feature forest. For example, if the disjunctive nodes (equivalence classes of individual entries) representing the categories NP and $S\backslash NP$ combine to produce a conjunctive node S, the resulting S node will have a verb-subject dependency associated with it.

In Figure 2, *cdeps*(c) is the number of dependencies on conjunctive node c which appear in partial structure G; *dmax*(c) is the maximum number of dependencies in G produced by any sub-derivation headed by c; *dmax*(d) is the same value for disjunctive node d. Recursive definitions for calculating these values are given; the base case occurs when conjunctive nodes have no disjunctive daughters.

The algorithm identifies all those root nodes heading derivations which are consistent with the partial dependency structure G, and traverses the chart top-down marking the nodes in those derivations. The insight behind the algorithm is that, for two conjunctive nodes in the same equivalence class, if one node heads a sub-derivation producing more dependencies in G than the other node, then the node with

[2]Note that the procedure does return *all* the gold-standard dependencies for some sentences.

less dependencies in G cannot be part of a derivation consistent with G.

The conjunctive and disjunctive nodes appearing in derivations consistent with G form a new "gold-standard" feature forest. The gold-standard forest, and the complete forest containing all derivations spanning the sentence, can be used to estimate the likelihood value and feature expectations required by the estimation algorithm. Let $E^{\Phi}_{\Lambda} f_i$ be the expected value of f_i over the forest Φ for model Λ; then the values in (5) can be obtained by calculating $E^{\Phi_j}_{\Lambda} f_i$ for the complete forest Φ_j for each sentence S_j in the training data (the second sum in (5)), and also $E^{\Psi_j}_{\Lambda} f_i$ for each forest Ψ_j of derivations consistent with the partial gold-standard dependency structure for sentence S_j (the first sum in (5)):

$$\frac{\partial L(\Lambda)}{\partial \lambda_i} = \sum_{j=1}^{m} (E^{\Psi_j}_{\Lambda} f_i - E^{\Phi_j}_{\Lambda} f_i) \qquad (6)$$

The likelihood in (4) can be calculated as follows:

$$L(\Lambda) = \sum_{j=1}^{m} (\log Z_{\Psi_j} - \log Z_{\Phi_j}) \qquad (7)$$

where $\log Z_{\Phi}$ is the normalisation constant for Φ.

4 Experiments

The resource used for the experiments is CCGbank (Hockenmaier, 2003), which consists of normal-form CCG derivations derived from the phrase-structure trees in the Penn Treebank. It also contains predicate-argument dependencies which we use for development and final evaluation.

4.1 Accuracy of Dependency Extraction

Sections 2-21 of CCGbank were used to investigate the accuracy of the partial dependency structures returned by the extraction procedure. Full, correct dependency structures for the sentences in 2-21 were created by running our CCG parser (Clark and Curran, 2004b) over the gold-standard derivation for each sentence, outputting the dependencies. This resulted in full dependency structures for 37,283 of the sentences in sections 2-21.

Table 1 gives precision and recall values for the dependencies obtained from the extraction procedure, for the 37,283 sentences for which we have complete dependency structures. The SentAcc column gives the percentage of training sentences for which the partial dependency structures are completely correct. For a given sentence, the extraction procedure returns all dependencies occurring in at least $k\%$ of the derivations licenced by the gold-standard lexical category sequence. The lexical category sequences for the sentences in 2-21 can easily be read off the CCGbank derivations.

k	Precision	Recall	SentAcc
0.99999	99.76	74.96	13.84
0.9	99.69	79.37	16.52
0.85	99.65	81.30	18.40
0.8	99.57	82.96	19.51
0.7	99.09	85.87	22.46
0.6	98.00	88.67	26.28

Table 1: Accuracy of the Partial Dependency Data

The derivations licenced by a lexical category sequence were created using the CCG parser described in Clark and Curran (2004b). The parser uses a small number of combinatory rules to combine the categories, along with the CKY chart-parsing algorithm described in Steedman (2000). It also uses some unary type-changing rules and punctuation rules obtained from the derivations in CCGbank.[3] The parser builds a packed representation, and counting the number of derivations in which a dependency occurs can be performed using a dynamic programming algorithm similar to the inside-outside algorithm.

Table 1 shows that, by varying the value of k, it is possible to get the recall of the extracted dependencies as high as 85.9%, while still maintaining a precision value of over 99%.

4.2 Accuracy of the Parser

The training data for the dependency model was created by first supertagging the sentences in sections 2-21, using the supertagger described in Clark and Curran (2004b).[4] The average number of categories

[3] Since our training method is intended to be applicable in the absence of derivation data, the use of such rules may appear suspect. However, we argue that the type-changing and punctuation rules could be manually created for a new domain by examining the lexical category data.

[4] An improved version of the supertagger was used for this paper in which the forward-backward algorithm is used to calculate the lexical category probability distributions.

assigned to each word is determined by a parameter, β, in the supertagger. A category is assigned to a word if the category's probability is within β of the highest probability category for that word.

For these experiments, we used a β value of 0.01, which assigns roughly 1.6 categories to each word, on average; we also ensured that the correct lexical category was in the set assigned to each word. (We did not do this when parsing the test data.) For some sentences, the packed charts can become very large. The supertagging approach we adopt for training differs to that used for testing: if the size of the chart exceeds some threshold, the value of β is increased, reducing ambiguity, and the sentence is supertagged and parsed again. The threshold which limits the size of the charts was set at 300 000 individual entries. Two further values of β were used: 0.05 and 0.1.

Packed charts were created for each sentence and stored in memory. It is essential that the packed charts for each sentence contain at least one derivation leading to the gold-standard dependency structure. Not all rule instantiations in CCGbank can be produced by our parser; hence it is not possible to produce the gold standard for every sentence in Sections 2-21. For the full-data model we used 34 336 sentences (86.7% of the total). For the partial-data models we were able to use slightly more, since the partial structures are easier to produce. Here we used 35,709 sentences ($k = 0.85$).

Since some of the packed charts are very large, we used an 18-node Beowulf cluster, together with a parallel version of the BFGS training algorithm. The training time and number of iterations to convergence were 172 minutes and 997 iterations for the full-data model, and 151 minutes and 861 iterations for the partial-data model (k = 0.85). Approximate memory usage in each case was 17.6 GB of RAM.

The dependency model uses the same set of features described in Clark and Curran (2004b): dependency features representing predicate-argument dependencies (with and without distance measures); rule instantiation features encoding the combining categories together with the result category (with and without a lexical head); lexical category features, consisting of word–category pairs at the leaf nodes; and root category features, consisting of headword–category pairs at the root nodes. Further generalised features for each feature type are formed by replacing words with their POS tags.

k	LP	LR	F	CatAcc
0.99999	85.80	84.51	85.15	93.77
0.9	85.86	84.51	85.18	93.78
0.85	**85.89**	**84.50**	**85.19**	**93.71**
0.8	85.89	84.45	85.17	93.70
0.7	85.52	84.07	84.79	93.72
0.6	84.99	83.70	84.34	93.65
FullData	**87.16**	**85.84**	**86.50**	**93.79**
Random	74.63	72.53	73.57	89.31

Table 2: Accuracy of the Parser on Section 00

Only features which occur more than once in the training data are included, except that the cutoff for the rule features is 10 or more and the counting is performed across all derivations licenced by the gold-standard lexical category sequences. The larger cutoff was used since the productivity of the grammar can lead to large numbers of these features. The dependency model has 548 590 features. In order to provide a fair comparison, the same feature set was used for the partial-data and full-data models.

The CCG parsing consists of two phases: first the supertagger assigns the most probable categories to each word, and then the small number of combinatory rules, plus the type-changing and punctuation rules, are used with the CKY algorithm to build a packed chart.[5] We use the method described in Clark and Curran (2004b) for integrating the supertagger with the parser: initially a small number of categories is assigned to each word, and more categories are requested if the parser cannot find a spanning analysis. The "maximum-recall" algorithm described in Clark and Curran (2004b) is used to find the highest scoring dependency structure.

Table 2 gives the accuracy of the parser on Section 00 of CCGbank, evaluated against the predicate-argument dependencies in CCGbank.[6] The table gives labelled precision, labelled recall and F-score, and lexical category accuracy. Numbers are given for the partial-data model with various values of k, and for the full-data model, which provides an up-

[5]Gold-standard POS tags from CCGbank were used for all the experiments in this paper.

[6]There are some dependency types produced by our parser which are not in CCGbank; these were ignored for evaluation.

	LP	LR	F	CatAcc
k = 0.85	86.21	85.01	85.60	93.90
FullData	87.50	86.37	86.93	94.01

Table 3: Accuracy of the Parser on Section 23

k	Precision	Recall	SentAcc
0.99999	99.71	80.16	17.48
0.9999	99.68	82.09	19.13
0.999	99.49	85.18	22.18
0.99	99.00	88.95	27.69
0.95	98.34	91.69	34.95
0.9	97.82	92.84	39.18

Table 4: Accuracy of the Partial Dependency Data using Inside-Outside Scores

per bound for the partial-data model. We also give a lower bound which we obtain by randomly traversing a packed chart top-down, giving equal probability to each conjunctive node in an equivalence class. The precision and recall figures are over those sentences for which the parser returned an analysis (99.27% of Section 00).

The best result is obtained for a k value of 0.85, which produces partial dependency data with a precision of 99.7 and a recall of 81.3. Interestingly, the results show that decreasing k further, which results in partial data with a higher recall and only a slight loss in precison, harms the accuracy of the parser. The Random result also dispels any suspicion that the partial-model is performing well simply because of the supertagger; clearly there is still much work to be done after the supertagging phase.

Table 3 gives the accuracy of the parser on Section 23, using the best performing partial-data model on Section 00. The precision and recall figures are over those sentences for which the parser returned an analysis (99.63% of Section 23). The results show that the partial-data model is only 1.3% F-score short of the upper bound.

4.3 Further Experiments with Inside-Outside

In a final experiment, we attempted to exploit the high accuracy of the partial-data model by using it to provide new training data. For each sentence in Section 2-21, we parsed the gold-standard lexical category sequences and used the best performing partial-data model to assign scores to each dependency in the packed chart. The score for a dependency was the sum of the probabilities of all derivations producing that dependency, which can be calculated using the inside-outside algorithm. (This is the score used by the maximum-recall parsing algorithm.) Partial dependency structures were then created by returning all dependencies whose score was above some threshold k, as before. Table 4 gives the accuracy of the data created by this procedure. Note how these values differ to those reported in Table 1.

We then trained the dependency model on this partial data using the same method as before. However, the peformance of the parser on Section 00 using these new models was below that of the previous best performing partial-data model for all values of k. We report this negative result because we had hypothesised that using a probability model to score the dependencies, rather than simply the number of derivations in which they occur, would lead to improved performance.

5 Conclusions

Our main result is that it is possible to train a CCG dependency model from lexical category sequences alone and still obtain parsing results which are only 1.3% worse in terms of labelled F-score than a model trained on complete data. This is a noteworthy result and demonstrates the significant amount of information encoded in CCG lexical categories.

The engineering implication is that, since the dependency model can be trained without annotating recursive structures, and only needs sequence information at the word level, then it can be ported rapidly to a new domain (or language) by annotating new sequence data in that domain.

One possible response to this argument is that, since the lexical category sequence contains so much syntactic information, then the task of annotating category sequences must be almost as labour intensive as annotating full derivations. To test this hypothesis fully would require suitable annotation tools and subjects skilled in CCG annotation, which we do not currently have access to.

However, there is some evidence that annotating category sequences can be done very efficiently. Clark et al. (2004) describes a porting experiment

in which a CCG parser is adapted for the question domain. The supertagger component of the parser is trained on questions annotated at the lexical category level only. The training data consists of over 1,000 annotated questions which took less than a week to create. This suggests, as a very rough approximation, that 4 annotators could annotate 40,000 sentences with lexical categories (the size of the Penn Treebank) in a few months.

Another advantage of annotating with lexical categories is that a CCG supertagger can be used to perform most of the annotation, with the human annotator only required to correct the mistakes made by the supertagger. An accurate supertagger can be bootstrapped quicky, leaving only a small number of corrections for the annotator. A similar procedure is suggested by Doran et al. (1997) for porting an LTAG grammar to a new domain.

We have a proposed a novel solution to the annotation bottleneck for statistical parsing which exploits the lexicalized nature of CCG, and may therefore be applicable to other lexicalized grammar formalisms such as LTAG.

References

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 and Ted Briscoe. 2002. High precision extraction of grammatical relations. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 134–140, Taipei, Taiwan.

David Chiang. 2000. Statistical parsing with an automatically-extracted Tree Adjoining Grammar. In *Proceedings of the 38th Meeting of the ACL*, pages 456–463, Hong Kong.

Stephen Clark and James R. Curran. 2003. Log-linear models for wide-coverage CCG parsing. In *Proceedings of the EMNLP Conference*, pages 97–104, Sapporo, Japan.

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.

Stephen Clark, Mark Steedman, and James R. Curran. 2004. Object-extraction and question-parsing using CCG. In *Proceedings of the EMNLP Conference*, pages 111–118, Barcelona, Spain.

C. Doran, B. Hockey, P. Hopely, J. Rosenzweig, A. Sarkar, B. Srinivas, F. Xia, A. Nasr, and O. Rambow. 1997. Maintaining the forest and burning out the underbrush in XTAG. In *Proceedings of the ENVGRAM Workshop*, Madrid, Spain.

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.

Rebbeca Hwa. 1999. Supervised grammar induction using training data with limited constituent information. In *Proceedings of the 37th Meeting of the ACL*, pages 73–79, University of Maryland, MD.

Matthew Lease and Eugene Charniak. 2005. Parsing biomedical literature. In *Proceedings of the Second International Joint Conference on Natural Language Processing (IJCNLP-05)*, Jeju Island, Korea.

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Sixth Workshop on Natural Language Learning*, pages 49–55, Taipei, Taiwan.

Yusuke Miyao and Jun'ichi Tsujii. 2002. Maximum entropy estimation for feature forests. In *Proceedings of the Human Language Technology Conference*, San Diego, CA.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2004. Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the Penn Treebank. In *Proceedings of the First International Joint Conference on Natural Language Processing (IJCNLP-04)*, pages 684–693, Hainan Island, China.

Jorge Nocedal and Stephen J. Wright. 1999. *Numerical Optimization*. Springer, New York, USA.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Meeting of the ACL*, pages 128–135, Newark, DE.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell III, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th Meeting of the ACL*, pages 271–278, Philadelphia, PA.

Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steve Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *Proceedings of the 11th Conference of the European Association for Computational Linguistics*, Budapest, Hungary.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA.

Effective Self-Training for Parsing

David McClosky, Eugene Charniak, and Mark Johnson
Brown Laboratory for Linguistic Information Processing (BLLIP)
Brown University
Providence, RI 02912
{dmcc|ec|mj}@cs.brown.edu

Abstract

We present a simple, but surprisingly effective, method of self-training a two-phase parser-reranker system using readily available unlabeled data. We show that this type of bootstrapping is possible for parsing when the bootstrapped parses are processed by a discriminative reranker. Our improved model achieves an f-score of 92.1%, an absolute 1.1% improvement (12% error reduction) over the previous best result for Wall Street Journal parsing. Finally, we provide some analysis to better understand the phenomenon.

1 Introduction

In parsing, we attempt to uncover the syntactic structure from a string of words. Much of the challenge of this lies in extracting the appropriate parsing decisions from textual examples. Given sufficient labelled data, there are several "supervised" techniques of training high-performance parsers (Charniak and Johnson, 2005; Collins, 2000; Henderson, 2004). Other methods are "semi-supervised" where they use some labelled data to annotate unlabeled data. Examples of this include self-training (Charniak, 1997) and co-training (Blum and Mitchell, 1998; Steedman et al., 2003). Finally, there are "unsupervised" strategies where no data is labeled and all annotations (including the grammar itself) must be discovered (Klein and Manning, 2002).

Semi-supervised and unsupervised methods are important because good labeled data is expensive, whereas there is no shortage of unlabeled data. While some domain-language pairs have quite a bit of labelled data (e.g. news text in English), many other categories are not as fortunate. Less unsupervised methods are more likely to be portable to these new domains, since they do not rely as much on existing annotations.

2 Previous work

A simple method of incorporating unlabeled data into a new model is *self-training*. In self-training, the existing model first labels unlabeled data. The newly labeled data is then treated as truth and combined with the actual labeled data to train a new model. This process can be iterated over different sets of unlabeled data if desired. It is not surprising that self-training is not normally effective: Charniak (1997) and Steedman et al. (2003) report either minor improvements or significant damage from using self-training for parsing. Clark et al. (2003) applies self-training to POS-tagging and reports the same outcomes. One would assume that errors in the original model would be amplified in the new model.

Parser adaptation can be framed as a semi-supervised or unsupervised learning problem. In parser adaptation, one is given annotated training data from a source domain and unannotated data from a target. In some cases, some annotated data from the target domain is available as well. The goal is to use the various data sets to produce a model that accurately parses the target domain data despite seeing little or no annotated data from that domain. Gildea (2001) and Bacchiani et al. (2006) show that out-of-domain training data can improve parsing ac-

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 152–159,
New York, June 2006.

curacy. The unsupervised adaptation experiment by Bacchiani et al. (2006) is the only successful instance of parsing self-training that we have found. Our work differs in that all our data is in-domain while Bacchiani et al. uses the Brown corpus as labelled data. These correspond to different scenarios. Additionally, we explore the use of a reranker.

Co-training is another way to train models from unlabeled data (Blum and Mitchell, 1998). Unlike self-training, co-training requires multiple learners, each with a different "view" of the data. When one learner is confident of its predictions about the data, we apply the predicted label of the data to the training set of the other learners. A variation suggested by Dasgupta et al. (2001) is to add data to the training set when multiple learners agree on the label. If this is the case, we can be more confident that the data was labelled correctly than if only one learner had labelled it.

Sarkar (2001) and Steedman et al. (2003) investigated using co-training for parsing. These studies suggest that this type of co-training is most effective when small amounts of labelled training data is available. Additionally, co-training for parsing can be helpful for parser adaptation.

3 Experimental Setup

Our parsing model consists of two phases. First, we use a generative parser to produce a list of the top n parses. Next, a discriminative reranker reorders the n-best list. These components constitute two views of the data, though the reranker's view is restricted to the parses suggested by the first-stage parser. The reranker is not able to suggest new parses and, moreover, uses the probability of each parse tree according to the parser as a feature to perform the reranking. Nevertheless, the reranker's value comes from its ability to make use of more powerful features.

3.1 The first-stage 50-best parser

The first stage of our parser is the lexicalized probabilistic context-free parser described in (Charniak, 2000) and (Charniak and Johnson, 2005). The parser's grammar is a smoothed third-order Markov grammar, enhanced with lexical heads, their parts of speech, and parent and grandparent information. The parser uses five probability distributions, one each for heads, their parts-of-speech, head-constituent, left-of-head constituents, and right-of-head constituents. As all distributions are conditioned with five or more features, they are all heavily backed off using Chen back-off (the *average-count* method from Chen and Goodman (1996)). Also, the statistics are lightly pruned to remove those that are statistically less reliable/useful. As in (Charniak and Johnson, 2005) the parser has been modified to produce n-best parses. However, the n-best parsing algorithm described in that paper has been replaced by the much more efficient algorithm described in (Jimenez and Marzal, 2000; Huang and Chang, 2005).

3.2 The MaxEnt Reranker

The second stage of our parser is a Maximum Entropy reranker, as described in (Charniak and Johnson, 2005). The reranker takes the 50-best parses for each sentence produced by the first-stage 50-best parser and selects the best parse from those 50 parses. It does this using the reranking methodology described in Collins (2000), using a Maximum Entropy model with Gaussian regularization as described in Johnson et al. (1999). Our reranker classifies each parse with respect to 1,333,519 features (most of which only occur on few parses). The features consist of those described in (Charniak and Johnson, 2005), together with an additional 601,577 features. These features consist of the parts-of-speech, possibly together with the words, that surround (i.e., precede or follow) the left and right edges of each constituent. The features actually used in the parser consist of all singletons and pairs of such features that have different values for at least one of the best and non-best parses of at least 5 sentences in the training data. There are 147,456 such features involving only parts-of-speech and 454,101 features involving parts-of-speech and words. These additional features are largely responsible for improving the reranker's performance on section 23 to 91.3% f-score (Charniak and Johnson (2005) reported an f-score of 91.0% on section 23).

3.3 Corpora

Our labeled data comes from the Penn Treebank (Marcus et al., 1993) and consists of about 40,000 sentences from Wall Street Journal (WSJ) articles

annotated with syntactic information. We use the standard divisions: Sections 2 through 21 are used for training, section 24 is held-out development, and section 23 is used for final testing. Our unlabeled data is the North American News Text corpus, NANC (Graff, 1995), which is approximately 24 million unlabeled sentences from various news sources. NANC contains no syntactic information. Sentence boundaries in NANC are induced by a simple discriminative model. We also perform some basic cleanups on NANC to ease parsing. NANC contains news articles from various news sources including the Wall Street Journal, though for this paper, we only use articles from the LA Times.

4 Experimental Results

We use the reranking parser to produce 50-best parses of unlabeled news articles from NANC. Next, we produce two sets of one-best lists from these 50-best lists. The parser-best and reranker-best lists represent the best parse for each sentence according to the parser and reranker, respectively. Finally, we mix a portion of parser-best or reranker-best lists with the standard Wall Street Journal training data (sections 2-21) to retrain a new parser (but not reranker[1]) model. The Wall Street Journal training data is combined with the NANC data in the following way: The count of each parsing event is the (optionally weighted) sum of the counts of that event in Wall Street Journal and NANC. Bacchiani et al. (2006) show that count merging is more effective than creating multiple models and calculating weights for each model (model interpolation). Intuitively, this corresponds to concatenating our training sets, possibly with multiple copies of each to account for weighting.

Some notes regarding evaluations: All numbers reported are f-scores[2]. In some cases, we evaluate only the parser's performance to isolate it from the reranker. In other cases, we evaluate the reranking parser as a whole. In these cases, we will use the term *reranking parser*.

Table 1 shows the difference in parser's (not reranker's) performance when trained on parser-best output versus reranker-best output. Adding parser-best sentences recreates previous self-training experiments and confirms that it is not beneficial. However, we see a large improvement from adding reranker-best sentences. One may expect to see a monotonic improvement from this technique, but this is not quite the case, as seen when we add 1,000k sentences. This may be due to some sections of NANC being less similar to WSJ or containing more noise. Another possibility is that these sections contains harder sentences which we cannot parse as accurately and thus are not as useful for self-training. For our remaining experiments, we will only use reranker-best lists.

Sentences added	Parser-best	Reranker-best
0 (baseline)	90.3	
50k	90.1	90.7
250k	90.1	90.7
500k	90.0	90.9
750k	89.9	91.0
1,000k	90.0	90.8
1,500k	90.0	90.8
2,000k	–	91.0

Table 1: f-scores after adding either parser-best or reranker-best sentences from NANC to WSJ training data. While the reranker was used to produce the reranker-best sentences, we performed this evaluation using only the first-stage parser to parse all sentences from section 22. We did not train a model where we added 2,000k parser-best sentences.

We also attempt to discover the optimal number of sentences to add from NANC. Much of the improvement comes from the addition of the initial 50,000 sentences, showing that even small amounts of new data can have a significant effect. As we add more data, it becomes clear that the maximum benefit to parsing accuracy by strictly adding reranker-best sentences is about 0.7% and that f-scores will asymptote around 91.0%. We will return to this when we consider the relative weightings of WSJ and NANC data.

One hypothesis we consider is that the reranked NANC data incorporated some of the features from the reranker. If this were the case, we would not see an improvement when evaluating a reranking parser

[1]We attempted to retrain the reranker using the self-trained sentences, but found no significant improvement.

[2]The harmonic mean of labeled precision (P) and labeled recall (R), i.e. $f = \frac{2 \times P \times R}{P+R}$

Sentences added	1	22	24
0 (baseline)	91.8	92.1	90.5
50k	91.8	92.4	90.8
250k	91.8	92.3	91.0
500k	92.0	92.4	90.9
750k	92.0	92.4	91.1
1,000k	92.1	92.2	91.3
1,500k	92.1	92.1	91.2
1,750k	92.1	92.0	91.3
2,000k	92.2	92.0	91.3

Table 2: f-scores from evaluating the reranking parser on three held-out sections after adding reranked sentences from NANC to WSJ training. These evaluations were performed on all sentences.

on the same models. In Table 2, we see that the new NANC data contains some information orthogonal to the reranker and improves parsing accuracy of the reranking parser.

Up to this point, we have only considered giving our true training data a relative weight of one. Increasing the weight of the Wall Street Journal data should improve, or at least not hurt, parsing performance. Indeed, this is the case for both the parser (figure not shown) and reranking parser (Figure 1). Adding more weight to the Wall Street Journal data ensures that the counts of our events will be closer to our more accurate data source while still incorporating new data from NANC. While it appears that the performance still levels off after adding about one million sentences from NANC, the curves corresponding to higher WSJ weights achieve a higher asymptote. Looking at the performance of various weights across sections 1, 22, and 24, we decided that the best combination of training data is to give WSJ a relative weight of 5 and use the first 1,750k reranker-best sentences from NANC.

Finally, we evaluate our new model on the test section of Wall Street Journal. In Table 3, we note that baseline system (i.e. the parser and reranker trained purely on Wall Street Journal) has improved by 0.3% over Charniak and Johnson (2005). The 92.1% f-score is the 1.1% absolute improvement mentioned in the abstract. The improvement from self-training is significant in both macro and micro tests ($p < 10^{-5}$).

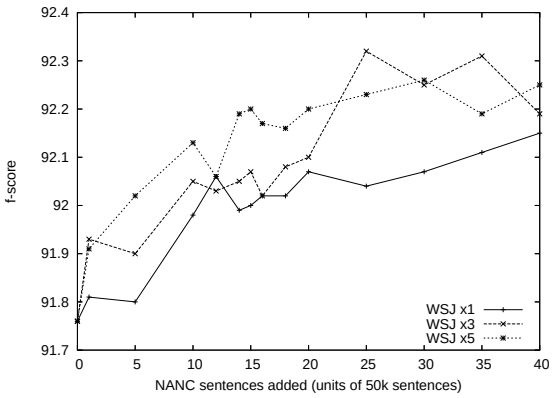


Figure 1: Effect of giving more relative weight to WSJ training data on reranking parser f-score. Evaluations were done from all sentences from section 1.

Model	f_{parser}	$f_{reranker}$
Charniak and Johnson (2005)	–	91.0
Current baseline	89.7	91.3
WSJ + NANC	91.0	92.1

Table 3: f-scores on WSJ section 23. f_{parser} and $f_{reranker}$ are the evaluation of the parser and reranking parser on all sentences, respectively. "WSJ + NANC" represents the system trained on WSJ training (with a relative weight of 5) and 1,750k sentences from the reranker-best list of NANC.

5 Analysis

We performed several types of analysis to better understand why the new model performs better. We first look at global changes, and then at changes at the sentence level.

5.1 Global Changes

It is important to keep in mind that while the reranker seems to be key to our performance improvement, the reranker per se never sees the extra data. It only sees the 50-best lists produced by the first-stage parser. Thus, the nature of the changes to this output is important.

We have already noted that the first-stage parser's one-best has significantly improved (see Table 1). In Table 4, we see that the 50-best oracle rate also im-

Model	1-best	10-best	50-best
Baseline	89.0	94.0	95.9
WSJ×1 + 250k	89.8	94.6	96.2
WSJ×5 + 1,750k	90.4	94.8	96.4

Table 4: Oracle f-scores of top n parses produced by baseline, a small self-trained parser, and the "best" parser

proves from 95.5% for the original first-stage parser, to 96.4% for our final model. We do not show it here, but if we self-train using first-stage one-best, there is no change in oracle rate.

The first-stage parser also becomes more "decisive." The average (geometric mean) of $\log_2$(Pr(1-best) / Pr(50th-best)) (i.e. the ratios between the probabilities in log space) increases from 11.959 for the baseline parser, to 14.104 for the final parser. We have seen earlier that this "confidence" is deserved, as the first-stage one-best is so much better.

5.2 Sentence-level Analysis

To this point we have looked at bulk properties of the data fed to the reranker. It has higher one best and 50-best-oracle rates, and the probabilities are more skewed (the higher probabilities get higher, the lows get lower). We now look at sentence-level properties. In particular, we analyzed the parsers' behavior on 5,039 sentences in sections 1, 22 and 24 of the Penn treebank. Specifically, we classified each sentence into one of three classes: those where the self-trained parser's f-score increased relative to the baseline parser's f-score, those where the f-score remained the same, and those where the self-trained parser's f-score decreased relative to the baseline parser's f-score. We analyzed the distribution of sentences into these classes with respect to four factors: sentence length, the number of unknown words (i.e., words not appearing in sections 2–21 of the Penn treebank) in the sentence, the number of coordinating conjunctions (CC) in the sentence, and the number of prepositions (IN) in the sentence. The distributions of classes (better, worse, no change) with respect to each of these factors individually are graphed in Figures 2 to 5.

Figure 2 shows how the self-training affects f-score as a function of sentence length. The top line

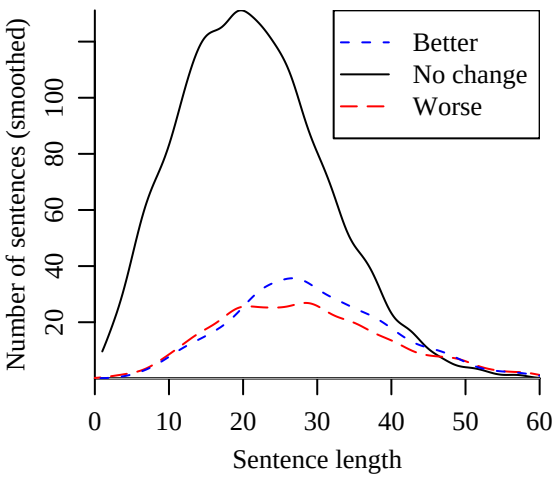


Figure 2: How self-training improves performance as a function of sentence length

shows that the f-score of most sentences remain unchanged. The middle line is the number of sentences that improved their f-score, and the bottom are those which got worse. So, for example, for sentences of length 30, about 80 were unchanged, 25 improved, and 22 worsened. It seems clear that there is no improvement for either very short sentences, or for very long ones. (For long ones the graph is hard to read. We show a regression analysis later in this section that confirms this statement.) While we did not predict this effect, in retrospect it seems reasonable. The parser was already doing very well on short sentences. The very long ones are hopeless, and the middle ones are just right. We call this the Goldilocks effect.

As for the other three of these graphs, their stories are by no means clear. Figure 3 seems to indicate that the number of unknown words in the sentence does *not* predict that the reranker will help. Figure 4 might indicate that the self-training parser improves prepositional-phrase attachment, but the graph looks suspiciously like that for sentence length, so the improvements might just be due to the Goldilocks effect. Finally, the improvement in Figure 5 is hard to judge.

To get a better handle on these effects we did a factor analysis. The factors we consider are number of CCs, INs, and unknowns, plus sentence length. As Figure 2 makes clear, the relative performance of the self-trained and baseline parsers does not

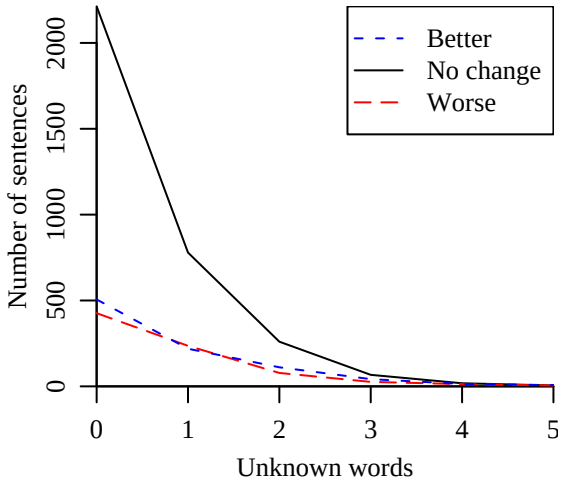


Figure 3: How self-training improves performance as a function of number of unknown words

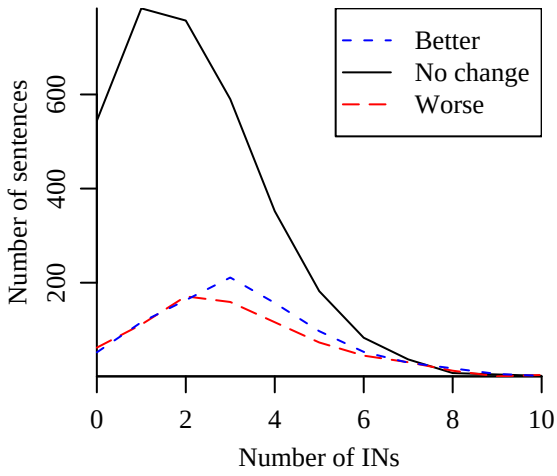


Figure 4: How self-training improves performance as a function of number of prepositions

	Estimate	Pr(>0)
(Intercept)	-0.25328	0.3649
BinnedLength(10,20]	0.02901	0.9228
BinnedLength(20,30]	0.45556	0.1201
BinnedLength(30,40]	0.40206	0.1808
BinnedLength(40,50]	0.26585	0.4084
BinnedLength(50,200]	-0.06507	0.8671
CCs	0.12333	0.0541

Table 5: Factor analysis for the question: does the self-trained parser improve the parse with the highest probability

vary linearly with sentence length, so we introduced binned sentence length (with each bin of length 10) as a factor.

Because the self-trained and baseline parsers produced equivalent output on 3,346 (66%) of the sentences, we restricted attention to the 1,693 sentences on which the self-trained and baseline parsers' f-scores differ. We asked the program to consider the following factors: binned sentence length, number of PPs, number of unknown words, and number of CCs. The results are shown in Table 5. The factor analysis is trying to model the log odds as a sum of linearly weighted factors. I.e,

$$log(P(1|x)/(1-P(1|x))) \quad = \quad \alpha_0 + \sum_{j=1}^{m} \alpha_j f_j(x)$$

In Table 5 the first column gives the name of the factor. The second the change in the log-odds resulting from this factor being present (in the case of CCs and INs, multiplied by the number of them) and the last column is the probability that this factor is really non-zero.

Note that there is no row for either PPs or unknown words. This is because we also asked the program to do a model search using the Akaike Information Criterion (AIC) over all single and pairwise factors. The model it chooses predicts that the self-trained parser is likely produce a better parse than the baseline only for sentences of length 20–40 or sentences containing several CCs. It did not include the number of unknown words and the number of INs as factors because they did not receive a weight significantly different from zero, and the AIC model search dropped them as factors from the model.

In other words, the self-trained parser is more likely to be correct for sentences of length 20–40 and as the number of CCs in the sentence increases. The self-trained parser does *not* improve prepositional-phrase attachment or the handling of unknown words.

This result is mildly perplexing. It is fair to say that neither we, nor anyone we talked to, thought conjunction handling would be improved. Conjunctions are about the hardest things in parsing, and we have no grip on exactly what it takes to help parse them. Conversely, everyone expected improvements on unknown words, as the self-training should dras-

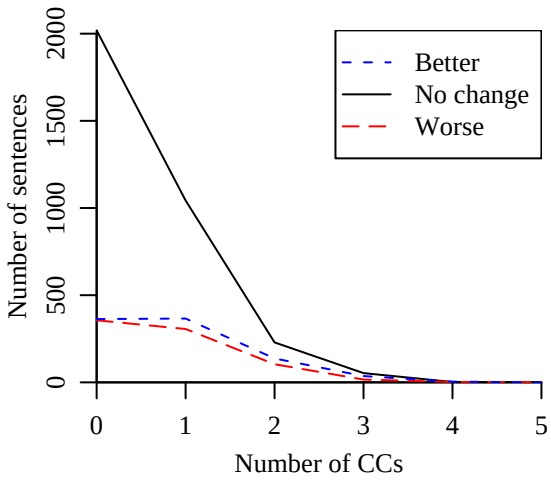


Figure 5: How self-training improves performance as a function of number of conjunctions

tically reduce the number of them. It is also the case that we thought PP attachment might be improved because of the increased coverage of preposition-noun and preposition-verb combinations that work such as (Hindle and Rooth, 1993) show to be so important.

Currently, our best conjecture is that unknowns are not improved because the words that are unknown in the WSJ are not significantly represented in the LA Times we used for self-training. CCs are difficult for parsers because each conjunct has only one secure boundary. This is particularly the case with longer conjunctions, those of VPs and Ss. One thing we know is that self-training always improves performance of the parsing model when used as a language model. We think CC improvement is connected with this fact and our earlier point that the probabilities of the 50-best parses are becoming more skewed. In essence the model is learning, in general, what VPs and Ss look like so it is becoming easier to pull them out of the stew surrounding the conjunct. Conversely, language modeling has comparatively less reason to help PP attachment. As long as the parser is doing it consistently, attaching the PP either way will work almost as well.

6 Conclusion

Contrary to received wisdom, self-training can improve parsing. In particular we have achieved an absolute improvement of 0.8% over the baseline performance. Together with a 0.3% improvement due to superior reranking features, this is a 1.1% improvement over the previous best parser results for section 23 of the Penn Treebank (from 91.0% to 92.1%). This corresponds to a 12% error reduction assuming that a 100% performance is possible, which it is not. The preponderance of evidence suggests that it is somehow the reranking aspect of the parser that makes this possible, but given no idea of why this should be, so we reserve final judgement on this matter.

Also contrary to expectations, the error analysis suggests that there has been no improvement in either the handing of unknown words, nor prepositional phrases. Rather, there is a general improvement in intermediate-length sentences (20-50 words), but no improvement at the extremes: a phenomenon we call the Goldilocks effect. The only specific syntactic phenomenon that seems to be affected is conjunctions. However, this is good news since conjunctions have long been considered the hardest of parsing problems.

There are many ways in which this research should be continued. First, the error analysis needs to be improved. Our tentative guess for why sentences with unknown words failed to improve should be verified or disproven. Second, there are many other ways to use self-trained information in parsing. Indeed, the current research was undertaken as the control experiment in a program to try much more complicated methods. We still have them to try: restricting consideration to more accurately parsed sentences as training data (sentence selection), trying to learn grammatical generalizations directly rather than simply including the data for training, etc.

Next there is the question of practicality. In terms of speed, once the data is loaded, the new parser is pretty much the same speed as the old — just under a second a sentence on average for treebank sentences. However, the memory requirements are largish, about half a gigabyte just to store the data. We are making our current best self-trained parser available[3] as machines with a gigabyte or more of RAM are becoming commonplace. Nevertheless, it would be interesting to know if the data can be pruned to

[3] `ftp://ftp.cs.brown.edu/pub/nlparser`

make the entire system less bulky.

Finally, there is also the nature of the self-trained data themselves. The data we use are from the LA Times. Those of us in parsing have learned to expect significant decreases in parsing accuracy even when moving the short distance from LA Times to Wall Street Journal. This seemingly has not occurred. Does this mean that the reranking parser somehow overcomes at least small genre differences? On this point, we have some pilot experiments that show great promise.

Acknowledgments

This work was supported by NSF grants LIS9720368, and IIS0095940, and DARPA GALE contract HR0011-06-2-0001. We would like to thank Michael Collins, Brian Roark, James Henderson, Miles Osborne, and the BLLIP team for their comments.

References

Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. 2006. MAP adaptation of stochastic grammars. *Computer Speech and Language*, 20(1):41–68.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT-98)*.

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 (ACL'05)*, pages 173–180, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, Menlo Park. AAAI Press/MIT Press.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *1st Annual Meeting of the NAACL*.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In Arivind Joshi and Martha Palmer, editors, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*.

Stephen Clark, James Curran, and Miles Osborne. 2003. Bootstrapping POS-taggers using unlabelled data. In *Proceedings of CoNLL-2003*.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Machine Learning: Proceedings of the 17th International Conference (ICML 2000)*, pages 175–182, Stanford, California.

Sanjoy Dasgupta, M.L. Littman, and D. McAllester. 2001. PAC generalization bounds for co-training. In *Advances in Neural Information Processing Systems (NIPS), 2001*.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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

James Henderson. 2004. Discriminative training of a neural network statistical parser. In *Proc. 42nd Meeting of Association for Computational Linguistics (ACL 2004), Barcelona, Spain*.

Donald Hindle and Mats Rooth. 1993. Structural ambiguity and lexical relations. *Computational Linguistics*, 19(1):103–120.

Liang Huang and David Chang. 2005. Better k-best parsing. Technical Report MS-CIS-05-08, Department of Computer Science, University of Pennsylvania.

Victor M. Jimenez and Andres Marzal. 2000. Computation of the n best parse trees for weighted and stochastic context-free grammars. In *Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition*. Springer LNCS 1876.

Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *The Proceedings of the 37th Annual Conference of the Association for Computational Linguistics*, pages 535–541, San Francisco. Morgan Kaufmann.

Dan Klein and Christopher Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting of the ACL*.

Michell 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.

Anoop Sarkar. 2001. Applying cotraining methods to statistical parsing. In *Proceedings of the 2001 NAACL Conference*.

Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *Proceedings of EACL 03*.

Multilingual Dependency Parsing using Bayes Point Machines

Simon Corston-Oliver
Microsoft Research
One Microsoft Way
Redmond, WA 98052
simonco@microsoft.com

Anthony Aue
Microsoft Research
One Microsoft Way
Redmond, WA 98052
anthaue@microsoft.com

Kevin Duh
Dept. of Electrical Eng.
Univ. of Washington
Seattle, WA 98195
duh@ee.washington.edu

Eric Ringger
Computer Science Dept.
Brigham Young Univ.
Provo, UT 84602
ringger@cs.byu.edu

Abstract

We develop dependency parsers for Arabic, English, Chinese, and Czech using Bayes Point Machines, a training algorithm which is as easy to implement as the perceptron yet competitive with large margin methods. We achieve results comparable to state-of-the-art in English and Czech, and report the first directed dependency parsing accuracies for Arabic and Chinese. Given the multilingual nature of our experiments, we discuss some issues regarding the comparison of dependency parsers for different languages.

1 Introduction

Dependency parsing is an alternative to constituency analysis with a venerable tradition going back at least two millenia. The last century has seen attempts to formalize dependency parsing, particularly in the Prague School approach to linguistics (Tesnière, 1959; Melčuk, 1988).

In a dependency analysis of syntax, words directly modify other words. Unlike constituency analysis, there are no intervening non-lexical nodes. We use the terms child and parent to denote the dependent term and the governing term respectively.

Parsing has many potential applications, ranging from question answering and information retrieval to grammar checking. Our intended application is machine translation in the Microsoft Research Treelet Translation System (Quirk et al., 2005; Menezes and Quirk, 2005). This system expects an analysis of the source language in which words are related by directed, unlabeled dependencies. For the purposes of developing machine translation for several language pairs, we are interested in dependency analyses for multiple languages.

The contributions of this paper are two-fold: First, we present a training algorithm called Bayes Point Machines (Herbrich et al., 2001; Harrington et al., 2003), which is as easy to implement as the perceptron, yet competitive with large margin methods. This algorithm has implications for anyone interested in implementing discriminative training methods for any application. Second, we develop parsers for English, Chinese, Czech, and Arabic and probe some linguistic questions regarding dependency analyses in different languages. To the best of our knowledge, the Arabic and Chinese results are the first reported results to date for directed dependencies. In the following, we first describe the data (Section 2) and the basic parser architecture (Section 3). Section 4 introduces the Bayes Point Machine while Section 5 describes the features for each language. We conclude with experimental results and discussions in Sections 6 and 7.

2 Data

We utilize publicly available resources in Arabic, Chinese, Czech, and English for training our dependency parsers.

For Czech we used the Prague Dependency Treebank version 1.0 (LDC2001T10). This is a corpus of approximately 1.6 million words. We divided the data into the standard splits for training, devel-

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 160–167,
New York, June 2006.

opment test and blind test. The Prague Czech Dependency Treebank is provided with human-edited and automatically-assigned morphological information, including part-of-speech labels. Training and evaluation was performed using the automatically-assigned labels.

For Arabic we used the Prague Arabic Dependency Treebank version 1.0 (LDC2004T23). Since there is no standard split of the data into training and test sections, we made an approximate 70%/15%/15% split for training/development test/blind test by sampling whole files. The Arabic Dependency Treebank is considerably smaller than that used for the other languages, with approximately 117,000 tokens annotated for morphological and syntactic relations. The relatively small size of this corpus, combined with the morphological complexity of Arabic and the heterogeneity of the corpus (it is drawn from five different newspapers across a three-year time period) is reflected in the relatively low dependency accuracy reported below. As with the Czech data, we trained and evaluated using the automatically-assigned part-of-speech labels provided with the data.

Both the Czech and the Arabic corpora are annotated in terms of syntactic dependencies. For English and Chinese, however, no corpus is available that is annotated in terms of dependencies. We therefore applied head-finding rules to treebanks that were annotated in terms of constituency.

For English, we used the Penn Treebank version 3.0 (Marcus et al., 1993) and extracted dependency relations by applying the head-finding rules of (Yamada and Matsumoto, 2003). These rules are a simplification of the head-finding rules of (Collins, 1999). We trained on sections 02-21, used section 24 for development test and evaluated on section 23. The English Penn Treebank contains approximately one million tokens. Training and evaluation against the development test set was performed using human-annotated part-of-speech labels. Evaluation against the blind test set was performed using part-of-speech labels assigned by the tagger described in (Toutanova et al., 2003).

For Chinese, we used the Chinese Treebank version 5.0 (Xue et al., 2005). This corpus contains approximately 500,000 tokens. We made an approximate 70%/15%/15% split for training/development test/blind test by sampling whole files. As with the English Treebank, training and evaluation against the development test set was performed using human-annotated part-of-speech labels. For evaluation against the blind test section, we used an implementation of the tagger described in (Toutanova et al., 2003). Trained on the same training section as that used for training the parser and evaluated on the development test set, this tagger achieved a token accuracy of 92.2% and a sentence accuracy of 63.8%.

The corpora used vary in homogeneity from the extreme case of the English Penn Treebank (a large corpus drawn from a single source, the Wall Street Journal) to the case of Arabic (a relatively small corpus–approximately 2,000 sentences–drawn from multiple sources). Furthermore, each language presents unique problems for computational analysis. Direct comparison of the dependency parsing results for one language to the results for another language is therefore difficult, although we do attempt in the discussion below to provide some basis for a more direct comparison. A common question when considering the deployment of a new language for machine translation is whether the natural language components available are of sufficient quality to warrant the effort to integrate them into the machine translation system. It is not feasible in every instance to do the integration work first and then to evaluate the output.

Table 1 summarizes the data used to train the parsers, giving the number of tokens (excluding traces and other empty elements) and counts of sentences.[1]

3 Parser Architecture

We take as our starting point a re-implementation of McDonald's state-of-the-art dependency parser (McDonald et al., 2005a). Given a sentence x, the goal of the parser is to find the highest-scoring parse $\hat{y}$ among all possible parses $y \in Y$:

$$\hat{y} = \arg\max_{y \in Y} s(x, y) \quad (1)$$

[1]The files in each partition of the Chinese and Arabic data are given at http://research.microsoft.com/~simonco/HLTNAACL2006.

Language	Total Tokens	Training Sentences	Development Sentences	Blind Sentences
Arabic	116,695	2,100	446	449
Chinese	527,242	14,735	1,961	2,080
Czech	1,595,247	73,088	7,319	7,507
English	1,083,159	39,832	1,346	2,416

Table 1: Summary of data used to train parsers.

For a given parse y, its score is the sum of the scores of all its dependency links $(i, j) \in y$:

$$s(x, y) = \sum_{(i,j)\in y} d(i, j) = \sum_{(i,j)\in y} \mathbf{w} \cdot \mathbf{f}(i, j) \quad (2)$$

where the link (i, j) indicates a head-child dependency between the token at position i and the token at position j. The score $d(i, j)$ of each dependency link (i, j) is further decomposed as the weighted sum of its features $\mathbf{f}(i, j)$.

This parser architecture naturally consists of three modules: (1) a decoder that enumerates all possible parses y and computes the argmax; (2) a training algorithm for adjusting the weights $\mathbf{w}$ given the training data; and (3) a feature representation $\mathbf{f}(i, j)$. Two decoders will be discussed here; the training algorithm and feature representation are discussed in the following sections.

A good decoder should satisfy several properties: ideally, it should be able to search through all valid parses of a sentence and compute the parse scores efficiently. Efficiency is a significant issue since there are usually an exponential number of parses for any given sentence, and the discriminative training methods we will describe later require repeated decoding at each training iteration. We reimplemented Eisner's decoder (Eisner, 1996), which searches among all *projective* parse trees, and the Chu-Liu-Edmonds' decoder (Chu and Liu, 1965; Edmonds, 1967), which searches in the space of both projective and non-projective parses. (A projective tree is a parse with no crossing dependency links.) For the English and Chinese data, the head-finding rules for converting from Penn Treebank analyses to dependency analyses creates trees that are guaranteed to be projective, so Eisner's algorithm suffices. For the Czech and Arabic corpora, a non-projective decoder is necessary. Both algorithms are $O(N^3)$, where N is the number of words in a sentence.[2] Refer to (McDonald et al., 2005b) for a detailed treatment of both algorithms.

4 Training: The Bayes Point Machine

In this section, we describe an *online* learning algorithm for training the weights $\mathbf{w}$. First, we argue why an online learner is more suitable than a batch learner like a Support Vector Machine (SVM) for this task. We then review some standard online learners (e.g. perceptron) before presenting the Bayes Point Machine (BPM) (Herbrich et al., 2001; Harrington et al., 2003).

4.1 Online Learning

An online learner differs from a batch learner in that it adjusts $\mathbf{w}$ incrementally as each input sample is revealed. Although the training data for our parsing problem exists as a batch (i.e. all input samples are available during training), we can apply online learning by presenting the input samples in some sequential order. For large training set sizes, a batch learner may face computational difficulties since there already exists an exponential number of parses per input sentence. Online learning is more tractable since it works with one input at a time.

A popular online learner is the perceptron. It adjusts $\mathbf{w}$ by updating it with the feature vector whenever a misclassification on the current input sample occurs. It has been shown that such updates converge in a finite number of iterations if the data is linearly separable. The averaged perceptron (Collins, 2002) is a variant which averages the $\mathbf{w}$ across all iterations; it has demonstrated good generalization especially with data that is not linearly separable, as in many natural language processing problems.

[2] The Chu-Liu-Edmonds' decoder, which is based on a maximal spanning tree algorithm, can run in $O(N^2)$, but our simpler implementation of $O(N^3)$ was sufficient.

Recently, the good generalization properties of Support Vector Machines have prompted researchers to develop large margin methods for the online setting. Examples include the margin perceptron (Duda et al., 2001), ALMA (Gentile, 2001), and MIRA (which is used to train the parser in (McDonald et al., 2005a)). Conceptually, all these methods attempt to achieve a large margin and approximate the maximum margin solution of SVMs.

4.2 Bayes Point Machines

The Bayes Point Machine (BPM) achieves good generalization similar to that of large margin methods, but is motivated by a very different philosophy of Bayesian learning or model averaging. In the Bayesian learning framework, we assume a prior distribution over $\mathbf{w}$. Observations of the training data revise our belief of $\mathbf{w}$ and produce a posterior distribution. The posterior distribution is used to create the final $\mathbf{w}_{\texttt{BPM}}$ for classification:

$$\mathbf{w}_{\texttt{BPM}} = E_{p(\mathbf{w}|D)}[\mathbf{w}] = \sum_{i=1}^{|V(D)|} p(\mathbf{w}_i|D)\ \mathbf{w}_i \quad (3)$$

where $p(\mathbf{w}|D)$ is the posterior distribution of the weights given the data D and $E_{p(\mathbf{w}|D)}$ is the expectation taken with respect to this distribution. The term $|V(D)|$ is the size of the *version space* $V(D)$, which is the set of weights $\mathbf{w}_i$ that is consistent with the training data (i.e. the set of $\mathbf{w}_i$ that classifies the training data with zero error). This solution achieves the so-called *Bayes Point*, which is the best approximation to the Bayes optimal solution given finite training data.

In practice, the version space may be large, so we approximate it with a finite sample of size I. Further, assuming a uniform prior over weights, we get the following equation:

$$\mathbf{w}_{\texttt{BPM}} = E_{p(\mathbf{w}|D)}[\mathbf{w}] \approx \sum_{i=1}^{I} \frac{1}{I}\mathbf{w}_i \quad (4)$$

Equation 4 can be computed by a very simple algorithm: (1) Train separate perceptrons on different random shuffles of the entire training data, obtaining a set of $\mathbf{w}_i$. (2) Take the average (arithmetic mean) of the weights $\mathbf{w}_i$. It is well-known that perceptron training results in different weight vector solutions if the data samples are presented sequentially in different orders. Therefore, random shuffles of the data and training a perceptron on each shuffle is effectively equivalent to sampling different models ($\mathbf{w}_i$) in the version space. Note that this averaging operation should not be confused with ensemble techniques such as Bagging or Boosting–ensemble techniques average the output hypotheses, whereas BPM averages the weights (models).

Input: Training set $D = ((x_1, y_1), (x_2, y_2), \ldots, (x_T, y_T))$
Output: $\mathbf{w}_{\texttt{BPM}}$

Initialize: $\mathbf{w}_{\texttt{BPM}} = \mathbf{0}$
for i = 1 to I; **do**
  Randomly shuffle the sequential order of samples in D
  Initialize: $\mathbf{w}_i = \mathbf{0}$
  for t = 1 to T; **do**
    $\hat{y}_t = \mathbf{w}_i \cdot x_t$
    if $(\hat{y}_t \neq y_t)$ **then** $\mathbf{w}_i = \mathbf{w}_i + y_t x_t$
  done
  $\mathbf{w}_{\texttt{BPM}} = \mathbf{w}_{\texttt{BPM}} + \frac{1}{I}\mathbf{w}_i$
done

Figure 1: Bayes Point Machine pseudo-code.

The BPM pseudocode is given in Figure 1. The inner loop is simply a perceptron algorithm, so the BPM is very simple and fast to implement. The outer loop is easily parallelizable, allowing speedups in training the BPM. In our specific implementation for dependency parsing, the line of the pseudocode corresponding to $[\hat{y}_t = \mathbf{w}_i \cdot x_t]$ is replaced by Eq. 1 and updates are performed for each incorrect dependency link. Also, we chose to average each individual perceptron (Collins, 2002) prior to Bayesian averaging.

Finally, it is important to note that the definition of the version space can be extended to include weights with non-zero training error, so the BPM can handle data that is not linearly separable. Also, although we only presented an algorithm for linear classifiers (parameterized by the weights), arbitrary kernels can be applied to BPM to allow non-linear decision boundaries. Refer to (Herbrich et al., 2001) for a comprehensive treatment of BPMs.

5 Features

Dependency parsers for all four languages were trained using the same set of feature types. The feature types are essentially those described in (McDonald et al., 2005a). For a given pair of tokens,

where one is hypothesized to be the parent and the other to be the child, we extract the word of the parent token, the part of speech of the parent token, the word of the child token, the part of speech of the child token and the part of speech of certain adjacent and intervening tokens. Some of these atomic features are combined in feature conjunctions up to four long, with the result that the linear classifiers described below approximate polynomial kernels. For example, in addition to the atomic features extracted from the parent and child tokens, the feature [ParentWord, ParentPOS, ChildWord, ChildPOS] is also added to the feature vector representing the dependency between the two tokens. Additional features are created by conjoining each of these features with the direction of the dependency (i.e. is the parent to the left or right of the child) and a quantized measure of the distance between the two tokens. Every token has exactly one parent. The root of the sentence has a special synthetic token as its parent.

Like McDonald et al, we add features that consider the first five characters of words longer than five characters. This truncated word crudely approximates stemming. For Czech and English the addition of these features improves accuracy. For Chinese and Arabic, however, it is clear that we need a different backoff strategy.

For Chinese, we truncate words longer than a single character to the first character.[3] Experimental results on the development test set suggested that an alternative strategy, truncation of words longer than two characters to the first two characters, yielded slightly worse results.

The Arabic data is annotated with gold-standard morphological information, including information about stems. It is also annotated with the output of an automatic morphological analyzer, so that researchers can experiment with Arabic without first needing to build these components. For Arabic, we truncate words to the stem, using the value of the lemma attribute.

All tokens are converted to lowercase, and numbers are normalized. In the case of English, Czech and Arabic, all numbers are normalized to a single token. In Chinese, months are normalized to a MONTH token, dates to a DATE token, years to a YEAR token. All other numbers are normalized to a single NUMBER token.

The feature types were instantiated using all oracle combinations of child and parent tokens from the training data. It should be noted that when the feature types are instantiated, we have considerably more features than McDonald et al. For example, for English we have 8,684,328 whereas they report 6,998,447 features. We suspect that this is mostly due to differences in implementation of the features that backoff to stems.

The averaged perceptrons were trained on the one-best parse, updating the perceptron for every edge and averaging the accumulated perceptrons after every sentence. Experiments in which we updated the perceptron based on k-best parses tended to produce worse results. The Chu-Liu-Edmonds algorithm was used for Czech. Experiments with the development test set suggested that the Eisner decoder gave better results for Arabic than the Chu-Liu-Edmonds decoder. We therefore used the Eisner decoder for Arabic, Chinese and English.

6 Results

Table 2 presents the accuracy of the dependency parsers. Dependency accuracy indicates for how many tokens we identified the correct head. Root accuracy, i.e. for how many sentences did we identify the correct root or roots, is reported as F1 measure, since sentences in the Czech and Arabic corpora can have multiple roots and since the parsing algorithms can identify multiple roots. Complete match indicates how many sentences were a complete match with the oracle dependency parse.

A convention appears to have arisen when reporting dependency accuracy to give results for English excluding punctuation (i.e., ignoring punctuation tokens in the output of the parser) and to report results for Czech including punctuation. In order to facilitate comparison of the present results with previously published results, we present measures including and excluding punctuation for all four languages. We hope that by presenting both sets of measurements, we also simplify one dimension along which published results of parse accuracy differ. A direct

[3] There is a near 1:1 correspondence between characters and morphemes in contemporary Mandarin Chinese. However, most content words consist of more than one morpheme, typically two.

	Including punctuation			Excluding punctuation		
Language	Dependency Accuracy	Root Accuracy	Complete Match	Dependency Accuracy	Root Accuracy	Complete Match
Arabic	79.9	90.0	9.80	79.8	87.8	10.2
Chinese	71.2	66.2	17.5	73.3	66.2	18.2
Czech	84.0	88.8	30.9	84.3	76.2	32.2
English	90.0	93.7	35.1	90.8	93.7	37.6

Table 2: Bayes Point Machine accuracy measured on blind test set.

comparison of parse results across languages is still difficult for reasons to do with the different nature of the languages, the corpora and the differing standards of linguistic detail annotated, but a comparison of parsers for two different languages where both results include punctuation is at least preferable to a comparison of results including punctuation to results excluding punctuation.

The results reported here for English and Czech are comparable to the previous best published numbers in (McDonald et al., 2005a), as Table 3 shows. This table compares McDonald et al.'s results for an averaged perceptron trained for ten iterations with no check for convergence (Ryan McDonald, pers. comm.), MIRA, a large margin classifier, and the current Bayes Point Machine results. To determine statistical significance we used confidence intervals for p=0.95. For the comparison of English dependency accuracy excluding punctuation, MIRA and BPM are both statistically significantly better than the averaged perceptron result reported in (McDonald et al., 2005a). MIRA is significantly better than BPM when measuring dependency accuracy and root accuracy, but BPM is significantly better when measuring sentences that match completely. From the fact that neither MIRA nor BPM clearly outperforms the other, we conclude that we have successfully replicated the results reported in (McDonald et al., 2005a) for English.

For Czech we also determined significance using confidence intervals for p=0.95 and compared results including punctuation. For both dependency accuracy and root accuracy, MIRA is statisticallty significantly better than averaged perceptron, and BPM is statistically significantly better than MIRA. Measuring the number of sentences that match completely, BPM is statistically significantly better than averaged perceptron, but MIRA is significantly better than BPM. Again, since neither MIRA nor BPM outperforms the other on all measures, we conclude that the results constitute a valiation of the results reported in (McDonald et al., 2005a).

For every language, the dependency accuracy of the Bayes Point Machine was greater than the accuracy of the best individual perceptron that contributed to that Bayes Point Machine, as Table 4 shows. As previously noted, when measuring against the development test set, we used human-annotated part-of-speech labels for English and Chinese.

Although the Prague Czech Dependency Treebank is much larger than the English Penn Treebank, all measurements are lower than the corresponding measurements for English. This reflects the fact that Czech has considerably more inflectional morphology than English, leading to data sparsity for the lexical features.

The results reported here for Arabic are, to our knowledge, the first published numbers for dependency parsing of Arabic. Similarly, the results for Chinese are the first published results for the dependency parsing of the Chinese Treebank 5.0.[4] Since the Arabic and Chinese numbers are well short of the numbers for Czech and English, we attempted to determine what impact the smaller corpora used for training the Arabic and Chinese parsers might have. We performed data reduction experiments, training the parsers on five random samples at each size smaller than the entire training set. Figure 2 shows the dependency accuracy measured on the complete development test set when training with samples of the data. The graph shows the average

[4](Wang et al., 2005) report numbers for *undirected* dependencies on the Chinese Treebank 3.0. We cannot meaningfully compare those numbers to the numbers here.

Language	Algorithm	DA	RA	CM
English (exc punc)	Avg. Perceptron	90.6	94.0	36.5
	MIRA	90.9	94.2	37.5
	Bayes Point Machine	90.8	93.7	37.6
Czech (inc punc)	Avg. Perceptron	82.9	88.0	30.3
	MIRA	83.3	88.6	31.3
	Bayes Point Machine	84.0	88.8	30.9

Table 3: Comparison to previous best published results reported in (McDonald et al., 2005a).

	Arabic	Chinese	Czech	English
Bayes Point Machine	78.4	83.8	84.5	91.2
Best averaged perceptron	77.9	83.1	83.5	90.8
Worst averaged perceptron	77.4	82.6	83.3	90.5

Table 4: Bayes Point Machine accuracy vs. averaged perceptrons, measured on development test set, excluding punctuation.

dependency accuracy for five runs at each sample size up to 5,000 sentences. English and Chinese accuracies in this graph use oracle part-of-speech tags. At all sample sizes, the dependency accuracy for English exceeds the dependency accuracy of the other languages. This difference is perhaps partly attributable to the use of oracle part-of-speech tags. However, we suspect that the major contributor to this difference is the part-of-speech tag set. The tags used in the English Penn Treebank encode traditional lexical categories such as noun, preposition, and verb. They also encode morphological information such as person (the VBZ tag for example is used for verbs that are third person, present tense–typically with the suffix -s), tense, number and degree of comparison. The part-of-speech tag sets used for the other languages encode lexical categories, but do not encode morphological information.[5] With small amounts of data, the perceptrons do not encounter sufficient instances of each lexical item to calculate reliable weights. The perceptrons are therefore forced to rely on the part-of-speech information.

It is surprising that the results for Arabic and Chinese should be so close as we vary the size of the training data (Figure 2) given that Arabic has rich morphology and Chinese very little. One possible explanation for the similarity in accuracy is that the rather poor root accuracy in Chinese indicates parses that have gone awry. Anecdotal inspection of parses suggests that when the root is not correctly identified, there are usually cascading related errors.

Czech, a morphologically complex language in which root identification is far from straightforward, exhibits the worst performance at small sample sizes. But (not shown) as the sample size increases, the accuracy of Czech and Chinese converge.

7 Conclusions

We have successfully replicated the state-of-the-art results for dependency parsing (McDonald et al., 2005a) for both Czech and English, using Bayes Point Machines. Bayes Point Machines have the appealing property of simplicity, yet are competitive with online wide margin methods.

We have also presented first results for dependency parsing of Arabic and Chinese, together with some analysis of the performance on those languages.

In future work we intend to explore the discriminative reranking of n-best lists produced by these parsers and the incorporation of morphological features.

[5]For Czech and Arabic we followed the convention established in previous parsing work on the Prague Czech Dependency Treebank of using the major and minor part-of-speech tags but ignoring other morphological information annotated on each node.

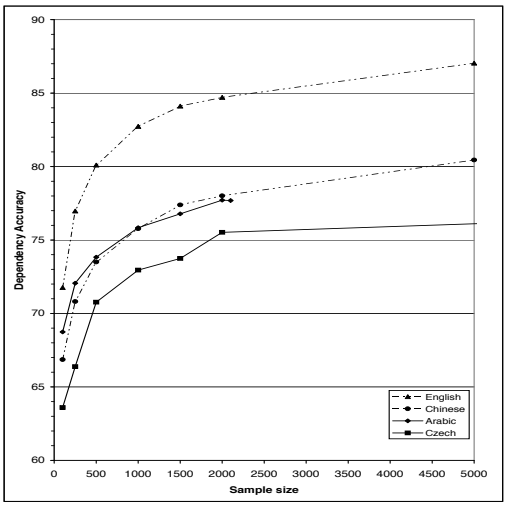


Figure 2: Dependency accuracy at various sample sizes. Graph shows average of five samples at each size and measures accuracy against the development test set.

Acknowledgements

We would like to thank Ryan McDonald, Otakar Smrž and Hiroyasu Yamada for help in various stages of the project.

References

Y.J. Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

Michael John Collins. 1999. *Head-Driven Statistical Models for Natural Language Processing*. Ph.D. thesis, University of Pennsylvania.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

R. O. Duda, P. E. Hart, and D. G. Stork. 2001. *Pattern Classification*. John Wiley & Sons, Inc.: New York.

J. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

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

Claudio Gentile. 2001. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242.

Edward Harrington, Ralf Herbrich, Jyrki Kivinen, John C. Platt, and Robert C. Williamson. 2003. Online bayes point machines. In *Proc. 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 241–252.

Ralf Herbrich, Thore Graepel, and Colin Campbell. 2001. Bayes point machines. *Journal of Machine Learning Research*, pages 245–278.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Assocation for Computational Linguistics*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005b. Online large-margin training of dependency parsers. Technical Report MS-CIS-05-11, Dept. of Computer and Information Science, Univ. of Pennsylvania.

Igor A. Melčuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press.

Arul Menezes and Chris Quirk. 2005. Microsoft research treelet translation system: IWSLT evaluation. In *Proceedings of the International Workshop on Spoken Language Translation*.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43rd annual meeting of the Association for Computational Linguistics*.

Lucien Tesnière. 1959. *Éléments de syntaxe structurale*. Librairie C. Klincksieck.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*, pages 252–259.

Qin Iris Wang, Dale Schuurmans, and Dekang Lin. 2005. Strictly lexical dependency parsing. In *Proceedings of the Ninth International Workshop on Parsing Technologies*, pages 152–159.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2).

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, pages 195–206.

Multilevel Coarse-to-fine PCFG Parsing

Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R. Shrivaths, Jeremy Moore, Michael Pozar, and Theresa Vu
Brown Laboratory for Linguistic Information Processing (BLLIP)
Brown University
Providence, RI 02912
ec@cs.brown.edu

Abstract

We present a PCFG parsing algorithm that uses a multilevel coarse-to-fine (mlctf) scheme to improve the efficiency of search for the best parse. Our approach requires the user to specify a sequence of nested partitions or equivalence classes of the PCFG nonterminals. We define a sequence of PCFGs corresponding to each partition, where the nonterminals of each PCFG are clusters of nonterminals of the original source PCFG. We use the results of parsing at a coarser level (i.e., grammar defined in terms of a coarser partition) to prune the next finer level. We present experiments showing that with our algorithm the work load (as measured by the total number of constituents processed) is decreased by a factor of ten with no decrease in parsing accuracy compared to standard CKY parsing with the original PCFG. We suggest that the search space over mlctf algorithms is almost totally unexplored so that future work should be able to improve significantly on these results.

1 Introduction

Reasonably accurate constituent-based parsing is fairly quick these days, if fairly quick means about a second per sentence. Unfortunately, this is still too slow for many applications. In some cases researchers need large quantities of parsed data and do not have the hundreds of machines necessary to parse gigaword corpora in a week or two. More pressingly, in real-time applications such as speech recognition, a parser would be only a part of a much larger system, and the system builders are not keen on giving the parser one of the ten seconds available to process, say, a thirty-word sentence. Even worse, some applications require the parsing of multiple candidate strings per sentence (Johnson and Charniak, 2004) or parsing from a lattice (Hall and Johnson, 2004), and in these applications parsing efficiency is even more important.

We present here a multilevel coarse-to-fine (mlctf) PCFG parsing algorithm that reduces the complexity of the search involved in finding the best parse. It defines a sequence of increasingly more complex PCFGs, and uses the parse forest produced by one PCFG to prune the search of the next more complex PCFG. We currently use four levels of grammars in our mlctf algorithm. The simplest PCFG, which we call the *level-0* grammar, contains only one nontrivial nonterminal and is so simple that minimal time is needed to parse a sentence using it. Nonetheless, we demonstrate that it identifies the locations of correct constituents of the parse tree (the "gold constituents") with high recall. Our level-1 grammar distinguishes only argument from modifier phrases (i.e., it has two nontrivial nonterminals), while our level-2 grammar distinguishes the four major phrasal categories (verbal, nominal, adjectival and prepositional phrases), and level 3 distinguishes all of the standard categories of the Penn treebank.

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 168–175,
New York, June 2006.

The nonterminal categories in these grammars can be regarded as clusters or equivalence classes of the original Penn treebank nonterminal categories. (In fact, we obtain these grammars by relabeling the node labels in the treebank and extracting a PCFG from this relabelled treebank in the standard fashion, but we discuss other approaches below.) We require that the partition of the nonterminals defined by the equivalence classes at level $l+1$ be a refinement of the partition defined at level l. This means that each nonterminal category at level $l+1$ is mapped to a unique nonterminal category at level l (although in general the mapping is many to one, i.e., each nonterminal category at level l corresponds to several nonterminal categories at level $l+1$).

We use the correspondence between categories at different levels to prune possible constituents. A constituent is considered at level $l+1$ only if the corresponding constituent at level l has a probability exceeding some threshold.. Thus parsing a sentence proceeds as follows. We first parse the sentence with the level-0 grammar to produce a parse forest using the CKY parsing algorithm. Then for each level $l+1$ we reparse the sentence with the level $l+1$ grammar using the level l parse forest to prune as described above. As we demonstrate, this leads to considerable efficiency improvements.

The paper proceeds as follows. We next discuss previous work (Section 2). Section 3 outlines the algorithm in more detail. Section 4 presents some experiments showing that the work load (as measured by the total number of constituents processed) is decreased by a factor of ten over standard CKY parsing at the final level. We also discuss some fine points of the results therein. Finally in section 5 we suggest that because the search space of mlctf algorithms is, at this point, almost totally unexplored, future work should be able to improve significantly on these results.

2 Previous Research

Coarse-to-fine search is an idea that has appeared several times in the literature of computational linguistics and related areas. The first appearance of this idea we are aware of is in Maxwell and Kaplan (1993), where a covering CFG is automatically extracted from a more detailed unification grammar and used to identify the possible locations of constituents in the more detailed parses of the sentence. Maxwell and Kaplan use their covering CFG to prune the search of their unification grammar parser in essentially the same manner as we do here, and demonstrate significant performance improvements by using their coarse-to-fine approach.

The basic theory of coarse-to-fine approximations and dynamic programming in a stochastic framework is laid out in Geman and Kochanek (2001). This paper describes the multilevel dynamic programming algorithm needed for coarse-to-fine analysis (which they apply to decoding rather than parsing), and show how to perform *exact coarse-to-fine* computation, rather than the heuristic search we perform here.

A paper closely related to ours is Goodman (1997). In our terminology, Goodman's parser is a two-stage ctf parser. The second stage is a standard tree-bank parser while the first stage is a regular-expression approximation of the grammar. Again, the second stage is constrained by the parses found in the first stage. Neither stage is smoothed. The parser of Charniak (2000) is also a two-stage ctf model, where the first stage is a smoothed Markov grammar (it uses up to three previous constituents as context), and the second stage is a lexicalized Markov grammar with extra annotations about parents and grandparents. The second stage explores all of the constituents not pruned out after the first stage. Related approaches are used in Hall (2004) and Charniak and Johnson (2005).

A quite different approach to parsing efficiency is taken in Caraballo and Charniak (1998) (and refined in Charniak et al. (1998)). Here efficiency is gained by using a standard chart-parsing algorithm and pulling constituents off the agenda according to (an estimate of) their probability given the sentence. This probability is computed by estimating Equation 1:

$$p(n^k_{i,j} \mid s) = \frac{\alpha(n^k_{i,j})\beta(n^k_{i,j})}{p(s)}. \qquad (1)$$

It must be estimated because during the bottom-up chart-parsing algorithm, the true outside probability cannot be computed. The results cited in Caraballo and Charniak (1998) cannot be compared directly to ours, but are roughly in the same equivalence class. Those presented in Charniak et al. (1998) are superior, but in Section 5 below we suggest that a combination of the techniques could yield better results still.

Klein and Manning (2003a) describe efficient A* for the most likely parse, where pruning is accomplished by using Equation 1 and a true upper bound on the outside probability. While their maximum is a looser estimate of the outside probability, it is an admissible heuristic and together with an A* search is guaranteed to find the best parse first. One question is if the guarantee is worth the extra search required by the looser estimate of the true outside probability.

Tsuruoka and Tsujii (2004) explore the framework developed in Klein and Manning (2003a), and seek ways to minimize the time required by the heap manipulations necessary in this scheme. They describe an iterative deepening algorithm that does not require a heap. They also speed computation by precomputing more accurate upper bounds on the outside probabilities of various kinds of constituents. They are able to reduce by half the number of constituents required to find the best parse (compared to CKY).

Most recently, McDonald et al. (2005) have implemented a dependency parser with good accuracy (it is almost as good at dependency parsing as Charniak (2000)) and very impressive speed (it is about ten times faster than Collins (1997) and four times faster than Charniak (2000)). It achieves its speed in part because it uses the Eisner and Satta (1999) algorithm for n^3 bilexical parsing, but also because dependency parsing has a much lower grammar constant than does standard PCFG parsing — after all, there are no phrasal constituents to consider. The current paper can be thought of as a way to take the sting out of the grammar constant for PCFGs by parsing first with very few phrasal constituents and adding them only after most constituents have been pruned away.

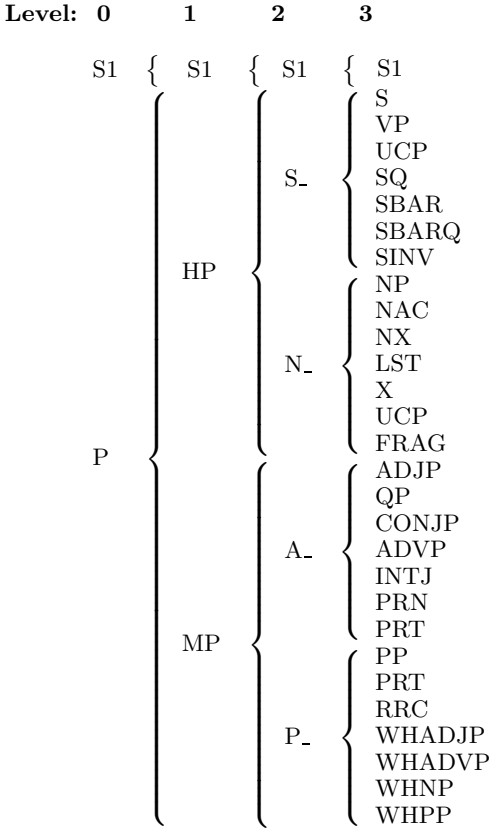


Figure 1: The levels of nonterminal labels

3 Multilevel Course-to-fine Parsing

We use as the underlying parsing algorithm a reasonably standard CKY parser, modified to allow unary branching rules.

The complete nonterminal clustering is given in Figure 1. We do not cluster preterminals. These remain fixed at all levels to the standard Penn-tree-bank set Marcus et al. (1993).

Level-0 makes two distinctions, the root node and everybody else. At level 1 we make one further distinction, between phrases that tend to be heads of constituents (NPs, VPs, and Ss) and those that tend to be modifiers (ADJPs, PPs, etc.). Level-2 has a total of five categories: root, things that are typically headed by nouns, those headed by verbs, things headed by prepositions, and things headed by classical modifiers (adjectives, adverbs, etc.). Finally, level 3 is the

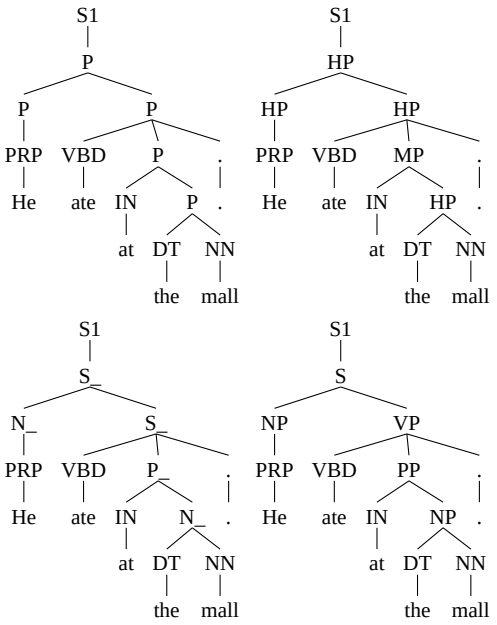


Figure 2: A tree represented at levels 0 to 3

classical tree-bank set. As an example, Figure 2 shows the parse for the sentence "He ate at the mall." at levels 0 to 3.

During training we create four grammars, one for each level of granularity. So, for example, at level 1 the tree-bank rule

S →NP VP .

would be translated into the rule

HP →HP HP .

That is, each constituent type found in "S →NP VP ." is mapped into its generalization at level 1. The probabilities of all rules are computed using maximum likelihood for constituents at that level.

The grammar used by the parser can best be described as being influenced by four components:

1. the nonterminals defined at that level of parsing,
2. the binarization scheme,
3. the generalizations defined over the binarization, and
4. extra annotation to improve parsing accuracy.

The first of these has already been covered. We discuss the other three in turn.

In anticipation of eventually lexicalizing the grammar we binarize from the head out. For example, consider the rule

A →a b c d e

where c is the head constituent. We binarize this as follows:

$A \rightarrow A_1$ e
$A_1 \rightarrow A_2$ d
$A_2 \rightarrow$a A_3
$A_3 \rightarrow$b c

Grammars induced in this way tend to be too specific, as the binarization introduce a very large number of very specialized phrasal categories (the A_i). Following common practice Johnson (1998; Klein and Manning (2003b) we Markovize by replacing these nonterminals with ones that remember less of the immediate rule context. In our version we keep track of only the parent, the head constituent and the constituent immediately to the right or left, depending on which side of the constituent we are processing. With this scheme the above rules now look like this:

$A \rightarrow A_{d,c}$ e
$A_{d,c} \rightarrow A_{a,c}$ d
$A_{a,c} \rightarrow$a $A_{b,c}$
$A_{b,c} \rightarrow$b c

So, for example, the rule "$A \rightarrow A_{d,c}$ e" would have a high probability if constituents of type A, with c as their head, often have d followed by e at their end.

Lastly, we add parent annotation to phrasal categories to improve parsing accuracy. If we assume that in this case we are expanding a rule for an A used as a child of Q, and a, b, c, d, and e are all phrasal categories, then the above rules become:

$A^Q \rightarrow A_{d,c}$ e^A
$A_{d,c} \rightarrow A_{a,c}$ d^A
$A_{a,c} \rightarrow a^A$ $A_{b,c}$
$A_{b,c} \rightarrow b^A$ c^A

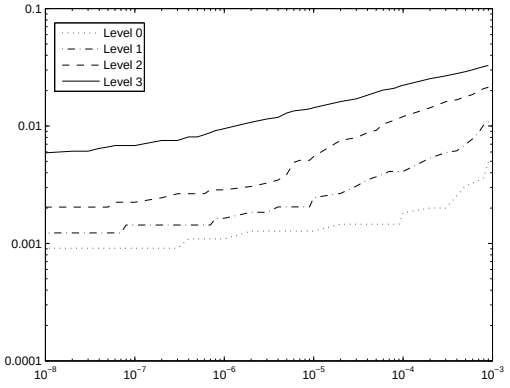


Figure 3: Probability of a gold constituent being pruned as a function of pruning thresholds for the first 100 sentences of the development corpus

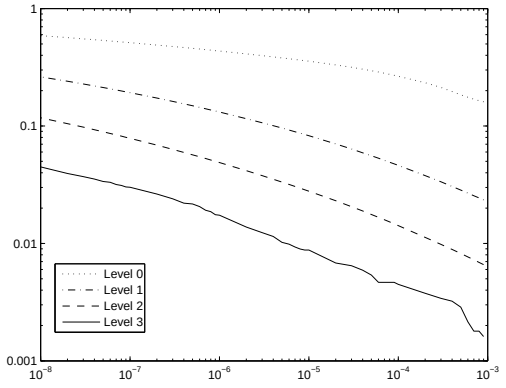


Figure 4: Fraction of incorrect constituents kept as a function of pruning thresholds for the first 100 sentences of the development corpus

Once we have parsed at a level, we evaluate the probability of a constituent $p(n^k_{i,j} \mid s)$ according to the standard inside-outside formula of Equation 1. In this equation $n^k_{i,j}$ is a constituent of type k spanning the words i to j, and $\alpha(\cdot)$ and $\beta(\cdot)$ are the outside and inside probabilities of the constituent, respectively. Because we prune at the end each granularity level, we can evaluate the equation exactly; no approximations are needed (as in, e.g., Charniak et al. (1998)).

During parsing, instead of building each constituent allowed by the grammar, we first test if the probability of the corresponding coarser constituent (which we have from Equation 1 in the previous round of parsing) is greater than a threshold. (The threshold is set empirically based upon the development data.) If it is below the threshold, we do not put the constituent in the chart. For example, before we can use a NP and a VP to create a S (using the rule above), we would first need to check that the probability in the coarser grammar of using the same span HP and HP to create a HP is above the threshold. We use the standard inside-outside formula to calculate this probability (Equation 1). The empirical results below justify our conjecture that there are thresholds that allow significant pruning while leaving the gold constituents untouched.

4 Results

In all experiments the system is trained on the Penn tree-bank sections 2-21. Section 23 is used for testing and section 24 for development. The input to the parser are the gold-standard parts of speech, not the words.

The point of parsing at multiple levels of granularity is to prune the results of rough levels before going on to finer levels. In particular, it is necessary for any pruning scheme to retain the true (gold-standard WSJ) constituents in the face of the pruning. To gain an idea of what is possible, consider Figure 3. According to the graph, at the zeroth level of parsing and a the pruning level 10^{-4} the probability that a gold constituent is deleted due to pruning is slightly more than 0.001 (or 0.1%). At level three it is slightly more that 0.01 (or 1.0%).

The companion figure, Figure 4 shows the retention rate of the non-gold (incorrect) constituents. Again, at pruning level 10^{-4} and parsing level 0 we retain about .3 (30%) of the bad constituents (so we pruned 70%), whereas at level 3 we retain about .004 (0.4%). Note that in the current paper we do not actually prune at level 3, instead return the Viterbi parse. We include pruning results here in anticipation of future work in which level 3 would be a precursor to still more fine-grained parsing.

As noted in Section 2, there is some (implicit)

Level	Constits Produced $*10^6$	Constits Pruned $*10^6$	% Pruned
0	8.82	7.55	86.5
1	9.18	6.51	70.8
2	11.2	9.48	84.4
3	11,8	0	0.0
total	40.4	–	–
3-only	392.0	0	0

Figure 5: Total constituents pruned at all levels for WSJ section 23, sentences of length ≤ 100

Level	Time for Level	Running Total
0	1598	1598
1	2570	4168
2	4303	8471
3	1527	9998
3-only	114654	–

Figure 6: Running times in seconds on WSJ section 23, with and without pruning

debate in the literature on using estimates of the outside probability in Equation 1, or instead computing the exact upper bound. The idea is that an exact upper bound gives one an admissible search heuristic but at a cost, since it is a less accurate estimator of the true outside probability. (Note that even the upper bound does not, in general, keep *all* of the gold constituents, since a non-perfect model will assign some of them low probability.) As is clear from Figure 3, the estimate works very well indeed.

On the basis of this graph, we set the lowest allowable constituent probability at $\geq 5 \cdot 10^{-4}$, $\geq 10^{-5}$, and $\geq 10^{-4}$ for levels 0,1, and 2, respectively. No pruning is done at level 3, since there is no level 4. After setting the pruning parameters on the development set we proceed to parse the test set (WSJ section 23). Figure 5 shows the resulting pruning statistics. The total number of constituents created at level 0, for all sentences combined, is $8.82 \cdot 10^6$. Of those $7.55 \cdot 10^6$ (or 86.5%) are pruned before going on to level 1. At level 1, the 1.3 million left over from level 0 expanded to a total of $9.18 \cdot 10^6$. 70.8% of these in turn are pruned, and so forth. The percent pruned at, e.g., level 1 in Figure 3 is much higher than that shown here because it considers *all* of the possible level-1 constituents, not just those left unpruned after level 0.

There is no pruning at level 3. There we simply return the Viterbi parse. We also show that with pruning we generate a total of $40.4 \cdot 10^6$ constituents. For comparison exhaustively parsing using the tree-bank grammar yields a total of $392 \cdot 10^6$ constituents. This is the factor-of-10 workload reduction mentioned in Section 1.

There are two points of interest. The first is that each level of pruning is worthwhile. We do not get most of the effect from one or the other level. The second point is that we get significant pruning at level 0. The reader may remember that level 0 distinguishes only between the root node and the rest. We initially expected that it would be too coarse to distinguish good from bad constituents at this level, but it proved as useful as the other levels. The explanation is that this level does use the full tree-bank preterminal tags, and in many cases these alone are sufficient to make certain constituents *very* unlikely. For example, what is the probability of *any* constituent of length two or greater ending in a preposition? The answer is: very low. Similarly for constituents of length two or greater ending in modal verbs, and determiners. Not quite so improbable, but nevertheless less likely than most, would be constituents ending in verbs, or ending just short of the end of the sentence.

Figure 6 shows how much time is spent at each level of the algorithm, along with a running total of the time spent to that point. (This is for all sentences in the test set, length ≤ 100.) The number for the unpruned parser is again about ten times that for the pruned version, but the number for the standard CKY version is probably too high. Because our CKY implementation is quite slow, we ran the unpruned version on many machines and summed the results. In all likelihood at least some of these machines were overloaded, a fact that our local job distributer would not notice. We suspect that the real number is significantly lower, though still

No pruning	77.9
With pruning	77.9
Klein and Manning (2003b)	77.4

Figure 7: Labeled precision/recall f-measure, WSJ section 23, all sentences of length ≤ 100

much higher than the pruned version.

Finally Figure 7 shows that our pruning is accomplished without loss of accuracy. The results with pruning include four sentences that did not receive any parses at all. These sentences received zeros for both precision and recall and presumably lowered the results somewhat. We allowed ourselves to look at the first of these, which turned out to contain the phrase:

(NP ... (INTJ (UH oh) (UH yes)) ...)

The training data does not include interjections consisting of two "UH"s, and thus a gold parse cannot be constructed. Note that a different binarization scheme (e.g. the one used in Klein and Manning (2003b) would have smoothed over this problem. In our case the unpruned version is able to patch together a lot of *very* unlikely constituents to produce a parse, but not a very good one. Thus we attribute the problem not to pruning, but to binarization.

We also show the results for the most similar Klein and Manning (2003b) experiment. Our results are slightly better. We attribute the difference to the fact that we have the gold tags and they do not, but their binarization scheme does not run into the problems that we encountered.

5 Conclusion and Future Research

We have presented a novel parsing algorithm based upon the coarse-to-fine processing model. Several aspects of the method recommend it. First, unlike methods that depend on best-first search, the method is "holistic" in its evaluation of constituents. For example, consider the impact of parent labeling. It has been repeatedly shown to improve parsing accuracy (Johnson, 1998; Charniak, 2000; Klein and Manning, 2003b), but it is difficult if not impossible to integrate with best-first search in bottom-up chart-parsing (as in Charniak et al. (1998)). The reason is that when working bottom up it is difficult to determine if, say, s^{sbar} is any more or less likely than s^{s}, as the evidence, working bottom up, is negligible. Since our method computes the exact outside probability of constituents (albeit at a coarser level) all of the top down information is available to the system. Or again, another very useful feature in English parsing is the knowledge that a constituent ends at the right boundary (minus punctuation) of a string. This can be included only in an ad-hoc way when working bottom up, but could be easily added here.

Many aspects of the current implementation that are far from optimal. It seems clear to us that extracting the maximum benefit from our pruning would involve taking the unpruned constituents and specifying them in all possible ways allowed by the next level of granularity. What we actually did is to propose all possible constituents at the next level, and immediately rule out those lacking a corresponding constituent remaining at the previous level. This was dictated by ease of implementation. Before using mlctf parsing in a production parser, the other method should be evaluated to see if our intuitions of greater efficiency are correct.

It is also possible to combine mlctf parsing with queue reordering methods. The best-first search method of Charniak et al. (1998) estimates Equation 1. Working bottom up, estimating the inside probability is easy (we just sum the probability of all the trees found to build this constituent). All the cleverness goes into estimating the outside probability. Quite clearly the current method could be used to provide a more accurate estimate of the outside probability, namely the outside probability at the coarser level of granularity.

There is one more future-research topic to add before we stop, possibly the most interesting of all. The particular tree of coarser to finer constituents that governs our mlctf algorithm (Figure 1) was created by hand after about 15 minutes of reflection and survives, except for typos, with only two modifications. There is no rea-

son to think it is anywhere close to optimal. It should be possible to define "optimal" formally and search for the best mlctf constituent tree. This would be a clustering problem, and, fortunately, one thing statistical NLP researchers know how to do is cluster.

Acknowledgments

This paper is the class project for Computer Science 241 at Brown University in fall 2005. The faculty involved were supported in part by DARPA GALE contract HR0011-06-2-0001. The graduate students were mostly supported by Brown University fellowships. The undergraduates were mostly supported by their parents. Our thanks to all.

References

Sharon Caraballo and Eugene Charniak. 1998. Figures of merit for best-first probabalistic parsing. *Computational Linguistics*, 24(2):275–298.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 2005 Meeting of the Association for Computational Linguistics*.

Eugene Charniak, Sharon Goldwater, and Mark Johnson. 1998. Edge-based best-first chart parsing. In *Proceedings of the Sixth Workshop on Very Large Corpora*, pages 127–133. Morgan Kaufmann.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*, pages 132–139.

Michael Collins. 1997. Three generative, lexicalized models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, San Francisco. Morgan Kaufmann.

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 457–464.

Stuart Geman and Kevin Kochanek. 2001. Dynamic programming and the representation of soft-decodable codes. *IEEE Transactions on Information Theory*, 47:549–568.

Joshua Goodman. 1997. Global thresholding and multiple-pass parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 1997)*.

Keith Hall and Mark Johnson. 2004. Attention shifting for parsing speech. In *The Proceedings of the 42th Annual Meeting of the Association for Computational Linguistics*, pages 40–46.

Keith Hall. 2004. *Best-first Word-lattice Parsing: Techniques for Integrated Syntactic Language Modeling*. Ph.D. thesis, Brown University.

Mark Johnson and Eugene Charniak. 2004. A TAG-based noisy-channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 33–39.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Dan Klein and Chris Manning. 2003a. A* parsing: Fast exact viterbi parse selection. In *Proceedings of HLT-NAACL'03*.

Dan Klein and Christopher Manning. 2003b. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*.

Michell 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.

John T. Maxwell and Ronald M. Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Linguistics*, 19(4):571–590.

Ryan McDonald, Toby Crammer, and Fernando Pereira. 2005. Online large margin training of dependency parsers. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics*.

Yoshimasa Tsuruoka and Jun'ichi Tsujii. 2004. Iterative cky parsing for probabilistic context-free grammars. In *International Joint Conference on Natural-Language Processing*.

A Fully-Lexicalized Probabilistic Model for Japanese Syntactic and Case Structure Analysis

Daisuke Kawahara[*] and **Sadao Kurohashi**[†]
Graduate School of Information Science and Technology, University of Tokyo
7-3-1 Hongo Bunkyo-ku, Tokyo, 113-8656, Japan
{kawahara,kuro}@kc.t.u-tokyo.ac.jp

Abstract

We present an integrated probabilistic model for Japanese syntactic and case structure analysis. Syntactic and case structure are simultaneously analyzed based on wide-coverage case frames that are constructed from a huge raw corpus in an unsupervised manner. This model selects the syntactic and case structure that has the highest generative probability. We evaluate both syntactic structure and case structure. In particular, the experimental results for syntactic analysis on web sentences show that the proposed model significantly outperforms known syntactic analyzers.

1 Introduction

Case structure (predicate-argument structure or logical form) represents what arguments are related to a predicate, and forms a basic unit for conveying the meaning of natural language text. Identifying such case structure plays an important role in natural language understanding.

In English, syntactic case structure can be mostly derived from word order. For example, the left argument of the predicate is the subject, and the right argument of the predicate is the object in most cases. Blaheta and Charniak proposed a statistical method for analyzing function tags in Penn Treebank, and achieved a really high accuracy of 95.7% for syntactic roles, such as SBJ (subject) and DTV (dative) (Blaheta and Charniak, 2000). In recent years, there have been many studies on semantic structure analysis (semantic role labeling) based on PropBank (Kingsbury et al., 2002) and FrameNet (Baker et al., 1998). These studies classify syntactic roles into semantic ones such as agent, experiencer and instrument.

Case structure analysis of Japanese is very different from that of English. In Japanese, postpositions are used to mark cases. Frequently used postpositions are "*ga*", "*wo*" and "*ni*", which usually mean nominative, accusative and dative. However, when an argument is followed by the topic-marking postposition "*wa*", its case marker is hidden. In addition, case-marking postpositions are often omitted in Japanese. These troublesome characteristics make Japanese case structure analysis very difficult.

To address these problems and realize Japanese case structure analysis, wide-coverage case frames are required. For example, let us describe how to apply case structure analysis to the following sentence:

bentou-wa	*taberu*
lunchbox-TM	eat
(eat lunchbox)	

In this sentence, *taberu* (eat) is a verb, and *bentou-wa* (lunchbox-TM) is a case component (i.e. argument) of *taberu*. The case marker of "*bentou-wa*" is hidden by the topic marker (TM) "*wa*". The analyzer matches "*bentou*" (lunchbox) with the most

[*] Currently, National Institute of Information and Communications Technology, JAPAN, dk@nict.go.jp

[†] Currently, Graduate School of Informatics, Kyoto University, kuro@i.kyoto-u.ac.jp

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 176–183,
New York, June 2006.

suitable case slot (CS) in the following case frame of "*taberu*" (eat).

	CS	examples
taberu	*ga*	person, child, boy, $\cdots$
	wo	lunch, lunchbox, dinner, $\cdots$

Since "*bentou*" (lunchbox) is included in "*wo*" examples, its case is analyzed as "*wo*". As a result, we obtain the case structure "ϕ:*ga bentou*:*wo taberu*", which means that "*ga*" (nominative) argument is omitted, and "*wo*" (accusative) argument is "*bentou*" (lunchbox). In this paper, we run such case structure analysis based on example-based case frames that are constructed from a huge raw corpus in an unsupervised manner.

Let us consider syntactic analysis, into which our method of case structure analysis is integrated. Recently, many accurate statistical parsers have been proposed (e.g., (Collins, 1999; Charniak, 2000) for English, (Uchimoto et al., 2000; Kudo and Matsumoto, 2002) for Japanese). Since they somehow use lexical information in the tagged corpus, they are called "lexicalized parsers". On the other hand, unlexicalized parsers achieved an almost equivalent accuracy to such lexicalized parsers (Klein and Manning, 2003; Kurohashi and Nagao, 1994). Accordingly, we can say that the state-of-the-art lexicalized parsers are mainly based on unlexical (grammatical) information due to the sparse data problem. Bikel also indicated that Collins' parser can use bilexical dependencies only 1.49% of the time; the rest of the time, it backs off to condition one word on just phrasal and part-of-speech categories (Bikel, 2004).

This paper aims at exploiting much more lexical information, and proposes a fully-lexicalized probabilistic model for Japanese syntactic and case structure analysis. Lexical information is extracted not from a small tagged corpus, but from a huge raw corpus as case frames. This model performs case structure analysis by a generative probabilistic model based on the case frames, and selects the syntactic structure that has the highest case structure probability.

2 Automatically Constructed Case Frames

We employ automatically constructed case frames (Kawahara and Kurohashi, 2002) for our model of case structure analysis. This section outlines the method for constructing the case frames.

Table 1: Case frame examples (examples are expressed only in English for space limitation.).

	CS	examples
youritsu (1) (support)	*ga*	<agent>, group, party, $\cdots$
	wo	<agent>, candidate, applicant
	ni	<agent>, district, election, $\cdots$
youritsu (2) (support)	*ga*	<agent>
	wo	<agent>, member, minister, $\cdots$
	ni	<agent>, candidate, successor
$\vdots$	$\vdots$	$\vdots$
itadaku (1) (have)	*ga*	<agent>
	wo	soup
itadaku (2) (be given)	*ga*	<agent>
	wo	advice, instruction, address
	kara	<agent>, president, circle, $\cdots$
$\vdots$	$\vdots$	$\vdots$

A large corpus is automatically parsed, and case frames are constructed from modifier-head examples in the resulting parses. The problems of automatic case frame construction are syntactic and semantic ambiguities. That is to say, the parsing results inevitably contain errors, and verb senses are intrinsically ambiguous. To cope with these problems, case frames are gradually constructed from reliable modifier-head examples.

First, modifier-head examples that have no syntactic ambiguity are extracted, and they are disambiguated by a couple of a verb and its closest case component. Such couples are explicitly expressed on the surface of text, and can be considered to play an important role in sentence meanings. For instance, examples are distinguished not by verbs (e.g., "*tsumu*" (load/accumulate)), but by couples (e.g., "*nimotsu-wo tsumu*" (load baggage) and "*keiken-wo tsumu*" (accumulate experience)). Modifier-head examples are aggregated in this way, and yield basic case frames.

Thereafter, the basic case frames are clustered to merge similar case frames. For example, since "*nimotsu-wo tsumu*" (load baggage) and "*busshi-wo tsumu*" (load supply) are similar, they are clustered. The similarity is measured using a thesaurus (Ikehara et al., 1997).

Using this gradual procedure, we constructed case frames from the web corpus (Kawahara and Kuro-

hashi, 2006). The case frames were obtained from approximately 470M sentences extracted from the web. They consisted of 90,000 verbs, and the average number of case frames for a verb was 34.3.

In Figure 1, some examples of the resulting case frames are shown. In this table, 'CS' means a case slot. <agent> in the table is a generalized example, which is given to the case slot where half of the examples belong to <agent> in a thesaurus (Ikehara et al., 1997). <agent> is also given to "*ga*" case slot that has no examples, because "*ga*" case components are usually agentive and often omitted.

3 Integrated Probabilistic Model for Syntactic and Case Structure Analysis

The proposed method gives a probability to each possible syntactic structure T and case structure L of the input sentence S, and outputs the syntactic and case structure that have the highest probability. That is to say, the system selects the syntactic structure T_{best} and the case structure L_{best} that maximize the probability $P(T, L|S)$:

$$\begin{aligned}(T_{best}, L_{best}) &= \underset{(T,L)}{\operatorname{argmax}} P(T, L|S) \\ &= \underset{(T,L)}{\operatorname{argmax}} \frac{P(T, L, S)}{P(S)} \\ &= \underset{(T,L)}{\operatorname{argmax}} P(T, L, S) \qquad (1)\end{aligned}$$

The last equation is derived because $P(S)$ is constant.

3.1 Generative Model for Syntactic and Case Structure Analysis

We propose a generative probabilistic model based on the dependency formalism. This model considers a clause as a unit of generation, and generates the input sentence from the end of the sentence in turn. $P(T, L, S)$ is defined as the product of a probability for generating a clause C_i as follows:

$$P(T, L, S) = \prod_{i=1..n} P(C_i|b_{h_i}) \qquad (2)$$

where n is the number of clauses in S, and b_{h_i} is C_i's modifying *bunsetsu*[1]. The main clause C_n at the end of a sentence does not have a modifying head, but we handle it by assuming b_{h_n} = EOS (End Of Sentence).

[1]In Japanese, *bunsetsu* is a basic unit of dependency, consisting of one or more content words and the following zero or more function words. It corresponds to a base phrase in English, and "*eojeol*" in Korean.

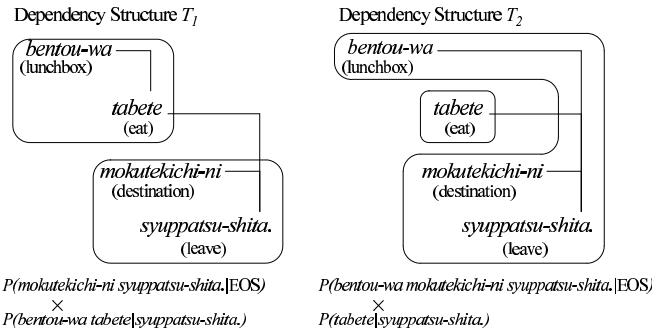


Figure 1: An Example of Probability Calculation.

For example, consider the sentence in Figure 1. There are two possible dependency structures, and for each structure the product of probabilities indicated below of the tree is calculated. Finally, the model chooses the highest-probability structure (in this case the left one).

C_i is decomposed into its predicate type f_i (including the predicate's inflection) and the rest case structure CS_i. This means that the predicate included in CS_i is lemmatized. *Bunsetsu* b_{h_i} is also decomposed into the content part w_{h_i} and the type f_{h_i}.

$$\begin{aligned}P(C_i|b_{h_i}) &= P(CS_i, f_i|w_{h_i}, f_{h_i}) \\ &= P(CS_i|f_i, w_{h_i}, f_{h_i})P(f_i|w_{h_i}, f_{h_i}) \\ &\approx P(CS_i|f_i, w_{h_i})P(f_i|f_{h_i}) \qquad (3)\end{aligned}$$

The last equation is derived because the content part in CS_i is independent of the type of its modifying head (f_{h_i}), and in most cases, the type f_i is independent of the content part of its modifying head (w_{h_i}).

For example, $P(\textit{bentou-wa tabete}|\textit{syuppatsu-shita})$ is calculated as follows:

$$P(CS(\textit{bentou-wa taberu})|\textit{te}, \textit{syuppatsu-suru})P(\textit{te}|\textit{ta.})$$

We call $P(CS_i|f_i, w_{h_i})$ *generative model for case structure* and $P(f_i|f_{h_i})$ *generative model for predicate type*. The following two sections describe these models.

3.2 Generative Model for Case Structure

We propose a generative probabilistic model of case structure. This model selects a case frame that

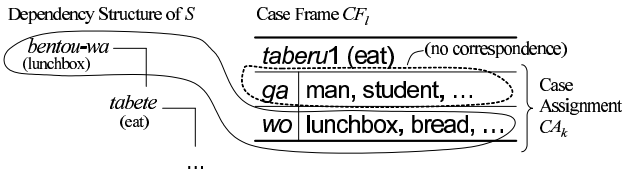


Figure 2: An example of case assignment CA_k.

matches the input case components, and makes correspondences between input case components and case slots.

A case structure CS_i consists of a predicate v_i, a case frame CF_l and a case assignment CA_k. Case assignment CA_k represents correspondences between input case components and case slots as shown in Figure 2. Note that there are various possibilities of case assignment in addition to that of Figure 2, such as corresponding "*bentou*" (lunchbox) with "*ga*" case. Accordingly, the index k of CA_k ranges up to the number of possible case assignments. By splitting CS_i into v_i, CF_l and CA_k, $P(CS_i|f_i, w_{h_i})$ is rewritten as follows:

$$\begin{aligned} P(CS_i|f_i, w_{h_i}) &= P(v_i, CF_l, CA_k|f_i, w_{h_i}) \\ &= P(v_i|f_i, w_{h_i}) \\ &\quad \times P(CF_l|f_i, w_{h_i}, v_i) \\ &\quad \times P(CA_k|f_i, w_{h_i}, v_i, CF_l) \\ &\approx P(v_i|w_{h_i}) \\ &\quad \times P(CF_l|v_i) \\ &\quad \times P(CA_k|CF_l, f_i) \end{aligned} \quad (4)$$

The above approximation is given because it is natural to consider that the predicate v_i depends on its modifying head w_{h_i}, that the case frame CF_l only depends on the predicate v_i, and that the case assignment CA_k depends on the case frame CF_l and the predicate type f_i.

The probabilities $P(v_i|w_{h_i})$ and $P(CF_l|v_i)$ are estimated from case structure analysis results of a large raw corpus. The remainder of this section illustrates $P(CA_k|CF_l, f_i)$ in detail.

3.2.1 Generative Probability of Case Assignment

Let us consider case assignment CA_k for each case slot s_j in case frame CF_l. $P(CA_k|CF_l, f_i)$ can be decomposed into the following product depending on whether a case slot s_j is filled with an input case component (content part n_j and type f_j) or vacant:

$$\begin{aligned} &P(CA_k|CF_l, f_i) = \\ &\quad \prod_{s_j:A(s_j)=1} P(A(s_j)=1, n_j, f_j|CF_l, f_i, s_j) \\ &\quad \times \prod_{s_j:A(s_j)=0} P(A(s_j)=0|CF_l, f_i, s_j) \\ &= \prod_{s_j:A(s_j)=1} \Big\{ P(A(s_j)=1|CF_l, f_i, s_j) \\ &\qquad \times P(n_j, f_j|CF_l, f_i, A(s_j)=1, s_j) \Big\} \\ &\quad \times \prod_{s_j:A(s_j)=0} P(A(s_j)=0|CF_l, f_i, s_j) \end{aligned} \quad (5)$$

where the function $A(s_j)$ returns 1 if a case slot s_j is filled with an input case component; otherwise 0.

$P(A(s_j) = 1|CF_l, f_i, s_j)$ and $P(A(s_j) = 0|CF_l, f_i, s_j)$ in equation (5) can be rewritten as $P(A(s_j) = 1|CF_l, s_j)$ and $P(A(s_j) = 0|CF_l, s_j)$, because the evaluation of case slot assignment depends only on the case frame. We call these probabilities *generative probability of a case slot*, and they are estimated from case structure analysis results of a large corpus.

Let us calculate $P(CS_i|f_i, w_{h_i})$ using the example in Figure 1. In the sentence, "*wa*" is a topic marking (TM) postposition, and hides the case marker. The generative probability of case structure varies depending on the case slot to which the topic marked phrase is assigned. For example, when a case frame of "*taberu*" (eat) $CF_{taberu1}$ with "*ga*" and "*wo*" case slots is used, $P(CS(\textit{bentou-wa taberu})|\textit{te}, \textit{syuppatsu-suru})$ is calculated as follows:

$$\begin{aligned} &P_1(CS(\textit{bentou-wa taberu})|\textit{te}, \textit{syuppatsu-suru}) = \\ &\quad P(\textit{taberu}|\textit{syuppatsu-suru}) \\ &\times \; P(CF_{\textit{taberu1}}|\textit{taberu}) \\ &\times \; P(\textit{bentou}, \textit{wa}|CF_{\textit{taberu1}}, \textit{te}, A(\textit{wo})=1, \textit{wo}) \\ &\times \; P(A(\textit{wo})=1|CF_{\textit{taberu1}}, \textit{wo}) \\ &\times \; P(A(\textit{ga})=0|CF_{\textit{taberu1}}, \textit{ga}) \end{aligned} \quad (6)$$

$$
\begin{aligned}
&P_2(CS(\textit{bentou-wa taberu})|\textit{te}, \textit{syupatsu-suru}) = \\
&\quad P(\textit{taberu}|\textit{syuppatsu-suru}) \\
&\times \quad P(CF_{\textit{taberu1}}|\textit{taberu}) \\
&\times \quad P(\textit{bentou}, \textit{wa}|CF_{\textit{taberu1}}, \textit{te}, A(\textit{ga}) = 1, \textit{ga}) \\
&\times \quad P(A(\textit{ga}) = 1|CF_{\textit{taberu1}}, \textit{ga}) \\
&\times \quad P(A(\textit{wo}) = 0|CF_{\textit{taberu1}}, \textit{wo}) \qquad (7)
\end{aligned}
$$

Such probabilities are computed for each case frame of "*taberu*" (eat), and the case frame and its corresponding case assignment that have the highest probability are selected.

We describe the generative probability of a case component $P(n_j, f_j|CF_l, f_i, A(s_j) = 1, s_j)$ below.

3.2.2 Generative Probability of Case Component

We approximate the generative probability of a case component, assuming that:

- a generative probability of content part n_j is independent of that of type f_j,
- and the interpretation of the surface case included in f_j does not depend on case frames.

Taking into account these assumptions, the generative probability of a case component is approximated as follows:

$$
\begin{aligned}
&P(n_j, f_j|CF_l, f_i, A(s_j) = 1, s_j) \approx \\
&\quad P(n_j|CF_l, A(s_j) = 1, s_j)\ P(f_j|s_j, f_i) \qquad (8)
\end{aligned}
$$

$P(n_j|CF_l, A(s_j) = 1, s_j)$ is the probability of generating a content part n_j from a case slot s_j in a case frame CF_l. This probability is estimated from case frames.

Let us consider $P(f_j|s_j, f_i)$ in equation (8). This is the probability of generating the type f_j of a case component that has a correspondence with the case slot s_j. Since the type f_j consists of a surface case c_j[2], a punctuation mark (comma) p_j and a topic marker "*wa*" t_j, $P(f_j|s_j, f_i)$ is rewritten as follows (using the chain rule):

$$
\begin{aligned}
P(f_j|s_j, f_i) &= P(c_j, t_j, p_j|s_j, f_i) \\
&= P(c_j|s_j, f_i) \\
&\quad \times P(p_j|s_j, f_i, c_j) \\
&\quad \times P(t_j|s_j, f_i, c_j, p_j) \\
&\approx P(c_j|s_j) \\
&\quad \times P(p_j|f_i) \\
&\quad \times P(t_j|f_i, p_j) \qquad (9)
\end{aligned}
$$

This approximation is given by assuming that c_j only depends on s_j, p_j only depends on f_j, and t_j depends on f_j and p_j. $P(c_j|s_j)$ is estimated from the Kyoto Text Corpus (Kawahara et al., 2002), in which the relationship between a surface case marker and a case slot is annotated by hand.

In Japanese, a punctuation mark and a topic marker are likely to be used when their belonging *bunsetsu* has a long distance dependency. By considering such tendency, f_i can be regarded as (o_i, u_i), where o_i means whether a dependent *bunsetsu* gets over another head candidate before its modifying head v_i, and u_i means a clause type of v_i. The value of o_i is binary, and u_i is one of the clause types described in (Kawahara and Kurohashi, 1999).

$$P(p_j|f_i) = P(p_j|o_i, u_i) \qquad (10)$$

$$P(t_j|f_i, p_j) = P(t_j|o_i, u_i, p_j) \qquad (11)$$

3.3 Generative Model for Predicate Type

Now, consider $P(f_i|f_{h_i})$ in the equation (3). This is the probability of generating the predicate type of a clause C_i that modifies b_{h_i}. This probability varies depending on the type of b_{h_i}.

When b_{h_i} is a predicate *bunsetsu*, C_i is a subordinate clause embedded in the clause of b_{h_i}. As for the types f_i and f_{h_i}, it is necessary to consider punctuation marks (p_i, p_{h_i}) and clause types (u_i, u_{h_i}). To capture a long distance dependency indicated by punctuation marks, o_{h_i} (whether C_i has a possible head candidate before b_{h_i}) is also considered.

$$P_{VBmod}(f_i|f_{h_i}) = P_{VBmod}(p_i, u_i|p_{h_i}, u_{h_i}, o_{h_i}) \qquad (12)$$

When b_{h_i} is a noun *bunsetsu*, C_i is an embedded clause in b_{h_i}. In this case, clause types and a punctuation mark of the modifiee do not affect the probability.

$$P_{NBmod}(f_i|f_{h_i}) = P_{NBmod}(p_i|o_{h_i}) \qquad (13)$$

[2] A surface case means a postposition sequence at the end of *bunsetsu*, such as "*ga*", "*wo*", "*koso*" and "*demo*".

Table 2: Data for parameter estimation.

probability	what is generated	data
$P(p_j\|o_i, u_j)$	punctuation mark	Kyoto Text Corpus
$P(t_j\|o_i, u_i, p_j)$	topic marker	Kyoto Text Corpus
$P(p_i, u_i\|p_{h_i}, u_{h_i}, o_{h_i})$	predicate type	Kyoto Text Corpus
$P(c_j\|s_j)$	surface case	Kyoto Text Corpus
$P(v_i\|w_{h_i})$	predicate	parsing results
$P(n_j\|CF_l, A(s_j) = 1, s_j)$	words	case frames
$P(CF_l\|v_i)$	case frame	case structure analysis results
$P(A(s_j) = \{0, 1\}\|CF_l, s_j)$	case slot	case structure analysis results

Table 3: Experimental results for syntactic analysis.

	baseline	proposed
all	3,447/3,976 (86.7%)	3,477/3,976 (87.4%)
NB→VB	1,310/1,547 (84.7%)	1,328/1,547 (85.8%)
TM	244/298 (81.9%)	242/298 (81.2%)
others	1,066/1,249 (85.3%)	1,086/1,249 (86.9%)
NB→NB	525/556 (94.4%)	526/556 (94.6%)
VB→VB	593/760 (78.0%)	601/760 (79.1%)
VB→NB	453/497 (91.1%)	457/497 (92.0%)

4 Experiments

We evaluated the syntactic structure and case structure outputted by our model. Each parameter is estimated using maximum likelihood from the data described in Table 2. All of these data are not existing or obtainable by a single process, but acquired by applying syntactic analysis, case frame construction and case structure analysis in turn. The process of case structure analysis in this table is a similarity-based method (Kawahara and Kurohashi, 2002). The case frames were automatically constructed from the web corpus comprising 470M sentences, and the case structure analysis results were obtained from 6M sentences in the web corpus.

The rest of this section first describes the experiments for syntactic structure, and then reports the experiments for case structure.

4.1 Experiments for Syntactic Structure

We evaluated syntactic structures analyzed by the proposed model. Our experiments were run on hand-annotated 675 web sentences [3]. The web sentences were manually annotated using the same criteria as the Kyoto Text Corpus. The system input was tagged automatically using the JUMAN morphological analyzer (Kurohashi et al., 1994). The syntactic structures obtained were evaluated with regard to dependency accuracy — the proportion of correct dependencies out of all dependencies except for the last dependency in the sentence end [4].

Table 3 shows the dependency accuracy. In the table, "baseline" means the rule-based syntactic parser, KNP (Kurohashi and Nagao, 1994), and "proposed" represents the proposed method. The proposed method significantly outperformed the baseline method (McNemar's test; $p < 0.05$). The dependency accuracies are classified into four types according to the *bunsetsu* classes (VB: verb *bunsetsu*, NB: noun *bunsetsu*) of a dependent and its head. The "NB→VB" type is further divided into two types: "TM" and "others". The type that is most related to case structure is "others" in "NB→VB". Its accuracy was improved by 1.6%, and the error rate was reduced by 10.9%. This result indicated that the proposed method is effective in analyzing dependencies related to case structure.

Figure 3 shows some analysis results, where the dotted lines represent the analysis by the baseline method, and the solid lines represent the analysis by the proposed method. Sentence (1) and (2) are incorrectly analyzed by the baseline but correctly analyzed by the proposed method.

There are two major causes that led to analysis errors.

Mismatch between analysis results and annotation criteria

In sentence (3) in Figure 3, the baseline method correctly recognized the head of "*iin-wa*" (commissioner-TM) as "*hirakimasu*" (open). However, the proposed method incorrectly judged it as "*oujite-imasuga*" (offer). Both analysis results can be considered to be correct semantically, but from

[3] The test set is not used for case frame construction and probability estimation.

[4] Since Japanese is head-final, the second last *bunsetsu* unambiguously depends on the last *bunsetsu*, and the last *bunsetsu* has no dependency.

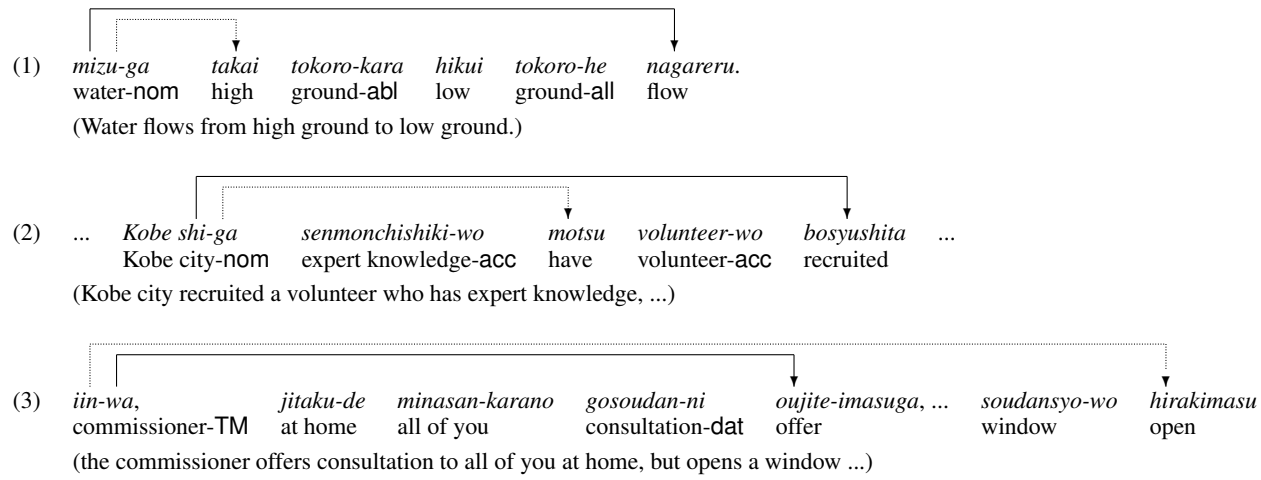


Figure 3: Examples of analysis results.

Table 4: Experimental results for case structure analysis.

	baseline	proposed
TM	72/105 (68.6%)	82/105 (78.1%)
clause	107/155 (69.0%)	121/155 (78.1%)

the viewpoint of our annotation criteria, the latter is not a syntactic relation, but an ellipsis relation. To address this problem, it is necessary to simultaneously evaluate not only syntactic relations but also indirect relations, such as ellipses and anaphora.

Linear weighting on each probability

We proposed a generative probabilistic model, and thus cannot optimize the weight of each probability. Such optimization could be a way to improve the system performance. In the future, we plan to employ a machine learning technique for the optimization.

4.2 Experiments for Case Structure

We applied case structure analysis to 215 web sentences which are manually annotated with case structure, and evaluated case markers of TM phrases and clausal modifiees by comparing them with the gold standard in the corpus. The experimental results are shown in table 4, in which the baseline refers to a similarity-based method (Kawahara and Kurohashi, 2002). The experimental results were really good compared to the baseline. It is difficult to compare the results with the previous work stated in the next section, because of different experimental settings (e.g., our evaluation includes parse errors in incorrect cases).

5 Related Work

There have been several approaches for syntactic analysis handling lexical preference on a large scale. Shirai et al. proposed a PGLR-based syntactic analysis method using large-scale lexical preference (Shirai et al., 1998). Their system learned lexical preference from a large newspaper corpus (articles of five years), such as $P(\text{pie}|wo, taberu)$, but did not deal with verb sense ambiguity. They reported 84.34% accuracy on 500 relatively short sentences from the Kyoto Text Corpus.

Fujio and Matsumoto presented a syntactic analysis method based on lexical statistics (Fujio and Matsumoto, 1998). They made use of a probabilistic model defined by the product of a probability of having a dependency between two cooccurring words and a distance probability. The model was trained on the EDR corpus, and performed with 86.89% accuracy on 10,000 sentences from the EDR corpus [5].

On the other hand, there have been a number of machine learning-based approaches using lexical preference as their features. Among these, Kudo and Matsumoto yielded the best performance (Kudo and Matsumoto, 2002). They proposed a chunking-based dependency analysis method using Support Vector Machines. They used two-fold cross validation on the Kyoto Text Corpus, and achieved 90.46%

[5]The evaluation includes the last dependencies in the sentence end, which are always correct.

accuracy [5]. However, it is very hard to learn sufficient lexical preference from several tens of thousands sentences of a hand-tagged corpus.

There has been some related work analyzing clausal modifiees and TM phrases. For example, Torisawa analyzed TM phrases using predicate-argument cooccurences and word classifications induced by the EM algorithm (Torisawa, 2001). Its accuracy was approximately 88% for "*wa*" and 84% for "*mo*". It is difficult to compare the accuracy of their system to ours, because the range of target expressions is different. Unlike related work, it is promising to utilize the resultant case frames for subsequent analyzes such as ellipsis or discourse analysis.

6 Conclusion

We have described an integrated probabilistic model for syntactic and case structure analysis. This model takes advantage of lexical selectional preference of large-scale case frames, and performs syntactic and case analysis simultaneously. The experiments indicated the effectiveness of our model. In the future, by incorporating ellipsis resolution, we will develop an integrated model of syntactic, case and ellipsis analysis.

References

Collin Baker, Charles Fillmore, and John Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics*, pages 86–90.

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

Don Blaheta and Eugene Charniak. 2000. Assigning function tags to parsed text. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 234–240.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 132–139.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Masakazu Fujio and Yuji Matsumoto. 1998. Japanese dependency structure analysis based on lexicalized statistics. In *Proceedings of the 3rd Conference on Empirical Methods in Natural Language Processing*, pages 88–96.

Satoru Ikehara, Masahiro Miyazaki, Satoshi Shirai, Akio Yokoo, Hiromi Nakaiwa, Kentarou Ogura, Yoshifumi Oyama, and Yoshihiko Hayashi, editors. 1997. *Japanese Lexicon*. Iwanami Publishing.

Daisuke Kawahara and Sadao Kurohashi. 1999. Corpus-based dependency analysis of Japanese sentences using verb bunsetsu transitivity. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium*, pages 387–391.

Daisuke Kawahara and Sadao Kurohashi. 2002. Fertilization of case frame dictionary for robust Japanese case analysis. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 425–431.

Daisuke Kawahara and Sadao Kurohashi. 2006. Case frame compilation from the web using high-performance computing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*.

Daisuke Kawahara, Sadao Kurohashi, and Kôiti Hasida. 2002. Construction of a Japanese relevance-tagged corpus. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, pages 2008–2013.

Paul Kingsbury, Martha Palmer, and Mitch Marcus. 2002. Adding semantic annotation to the Penn TreeBank. In *Proceedings of the Human Language Technology Conference*.

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

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proceedings of the Conference on Natural Language Learning*, pages 29–35.

Sadao Kurohashi and Makoto Nagao. 1994. A syntactic analysis method of long Japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20(4):507–534.

Sadao Kurohashi, Toshihisa Nakamura, Yuji Matsumoto, and Makoto Nagao. 1994. Improvements of Japanese morphological analyzer JUMAN. In *Proceedings of the International Workshop on Sharable Natural Language*, pages 22–28.

Kiyoaki Shirai, Kentaro Inui, Takenobu Tokunaga, and Hozumi Tanaka. 1998. An empirical evaluation on statistical parsing of Japanese sentences using lexical association statistics. In *Proceedings of the 3rd Conference on Empirical Methods in Natural Language Processing*, pages 80–87.

Kentaro Torisawa. 2001. An unsupervised method for canonicalization of Japanese postpositions. In *Proceedings of the 6th Natural Language Processing Pacific Rim Simposium*, pages 211–218.

Kiyotaka Uchimoto, Masaki Murata, Satoshi Sekine, and Hitoshi Isahara. 2000. Dependency model using posterior context. In *Proceedings of the 6th International Workshop on Parsing Technology*, pages 321–322.

Fully Parsing the Penn Treebank

Ryan Gabbard **Mitchell Marcus**
Department of
Computer and Information Science
University of Pennsylvania
{gabbard,mitch}@cis.upenn.edu

Seth Kulick
Institute for Research in
Cognitive Science
University of Pennsylvania
skulick@cis.upenn.edu

Abstract

We present a two stage parser that recovers Penn Treebank style syntactic analyses of new sentences including skeletal syntactic structure, and, for the first time, both function tags and empty categories. The accuracy of the first-stage parser on the standard Parseval metric matches that of the (Collins, 2003) parser on which it is based, despite the data fragmentation caused by the greatly enriched space of possible node labels. This first stage simultaneously achieves near state-of-the-art performance on recovering function tags with minimal modifications to the underlying parser, modifying less than ten lines of code. The second stage achieves state-of-the-art performance on the recovery of empty categories by combining a linguistically-informed architecture and a rich feature set with the power of modern machine learning methods.

1 Introduction

The trees in the Penn Treebank (Bies et al., 1995) are annotated with a great deal of information to make various aspects of the predicate-argument structure easy to decode, including both function tags and markers of "empty" categories that represent displaced constituents. Modern statistical parsers such as (Collins, 2003) and (Charniak, 2000) however ignore much of this information and return only an impoverished version of the trees. While there has been some work in the last few years on enriching the output of state-of-the-art parsers that output Penn Treebank-style trees with function tags (e.g. (Blaheta, 2003)) or empty categories (e.g. (Johnson, 2002; Dienes and Dubey, 2003a; Dienes and Dubey, 2003b), only one system currently available, the dependency graph parser of (Jijkoun and de Rijke, 2004), recovers some representation of both these aspects of the Treebank representation; its output, however, cannot be inverted to recover the original tree structures. We present here a parser,[1] the first we know of, that recovers full Penn Treebank-style trees. This parser uses a minimal modification of the Collins parser to recover function tags, and then uses this enriched output to achieve or better state-of-the-art performance on recovering empty categories.

We focus here on Treebank-style output for two reasons: First, annotators developing additional treebanks in new genres of English that conform to the Treebank II style book (Bies et al., 1995) must currently add these additional annotations by hand, a much more laborious process than correcting parser output (the currently used method for annotating the skeletal structure itself). Our new parser is now in use in a new Treebank annotation effort. Second, the accurate recovery of semantic structure from parser output requires establishing the equivalent of the information encoded within these representations.

Our parser consists of two components. The first-stage is a modification of Bikel's implementation (Bikel, 2004) of Collins' Model 2 that recovers function tags while parsing. Remarkably little modification to the parser is needed to allow it to produce function tags as part of its output, yet without decreasing the regular Parseval metric. While it is difficult to evaluate function tag assignment in isola-

We would like to thank Fernando Pereira, Dan Bikel, Tony Kroch and Mark Liberman for helpful suggestions. This work was supported in part under the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-C-0022, and in part by the National Science Foundation under grants NSF IIS-0520798 and NSF EIA 02-05448 and under an NSF graduate fellowship.

[1] The parser consists of two boxes; those who prefer to label it by its structure, as opposed to what it does, might call it a *parsing system*.

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 184–191,
New York, June 2006.

```
(S
  (NP-SBJ (DT The) (NN luxury)
          (NN auto) (NN maker) )
  (NP-TMP (JJ last) (NN year) )
  (VP (VBD sold)
    (NP (CD 1,214) (NNS cars) )
    (PP-LOC (IN in)
      (NP (DT the) (NNP U.S.) ))))
```

Figure 1: Example Tree

tion across the output of different parsers, our results match or exceed all but the very best of earlier tagging results, even though this earlier work is far more complicated than ours. The second stage uses a cascade of statistical classifiers which recovers the most important empty categories in the Penn Treebank style. These classifiers utilize a wide range of features, including crucially the function tags recovered in the first stage of parsing.

2 Motivation

Function tags are used in the current Penn Treebanks to augment nonterminal labels for various syntactic and semantic roles (Bies et al., 1995). For example, in Figure 1, `-SBJ` indicates the subject, `-TMP` indicates that the `NP last year` is serving as a temporal modifier, and `-LOC` indicates that the `PP` is specifying a location. Note that without these tags, it is very difficult to determine which of the two NPs directly dominated by S is in fact the subject. There are twenty function tags in the Penn Treebank, and following (Blaheta, 2003), we collect them into the five groups shown in Figure 2. While nonterminals can be assigned tags from different groups, they do not receive more than one tag from within a group. The Syntactic and Semantic groups are by far the most common tags, together making up over 90% of the function tag instances in the Penn Treebank.

Certain non–local dependencies must also be included in a syntactic analysis if it is to be most useful for recovering the predicate–argument structure of a complex sentence. For instance, in the sentence "The dragon I am trying to slay is green," it is important to know that *I* is the semantic subject and *the dragon* the semantic object of the slaying. The Penn Treebank (Bies et al., 1995) represents such dependencies by including nodes with no overt content (empty categories) in parse trees. In this work, we consider the three most frequent[2] and semantically important types of empty category annotations in most Treebank genres:

Null complementizers are denoted by the symbol `0`. They typically appear in places where, for example, an optional *that* or *who* is missing: "The king said `0` he could go." or "The man `(0)` I saw."

Traces of *wh*–movement are denoted by `*T*`, such as the noun phrase trace in "What$_1$ do you want `(NP *T*-1)`?" Note that *wh*–traces are co–indexed with their antecedents.

`(NP *)`s are used for several purposes in the Penn Treebank. Among the most common are passivization "`(NP-1` I) was captured `(NP *-1)`," and control "`(NP-1` I) tried `(NP *-1)` to get the best results."

Under this representation the above sentence would look like "(NP-1 The dragon) 0 (NP-2 I) am trying (NP *-2) to slay (NP *T*-1) is green."

Despite their importance, these annotations have largely been ignored in statistical parsing work. The importance of returning this information for most real applications of parsing has been greatly obscured by the Parseval metric (Black et al., 1991), which explicitly ignores both function tags and null elements. Because much statistical parsing research has been driven until recently by this metric, which has never been updated, the crucial role of parsing in recovering semantic structure has been generally ignored. An early exception to this was (Collins, 1997) itself, where Model 2 used function tags during the training process for heuristics to identify arguments (e.g., the `TMP` tag on the `NP` in Figure 1 disqualifies the `NP-TMP` from being treated as an argument). However, after this use, the tags are ignored, not included in the models, and absent from the parser output. Collins' Model 3 attempts to recover traces of *W*h-movement, with limited success.

3 Function Tags: Approach

Our system for restoring function tags is a modification of Collins' Model 2. We use the (Bikel, 2004)

[2]Excepting empty units (e.g. "$ 1,000,000 *U*"), which are not very interesting. According to Johnson, `(NP *)`s occur 28,146 times in the training portion of the PTB, `(NP *T*)`s occur 8,620 times, `0`s occur 7,969 times, and `(ADVP *T*)`s occur 2,492 times. In total, the types we consider cover roughly 84% of all the instances of empty categories in the training corpus.

Syntactic (55.9%)		Semantic (36.4%)				Misc (1.0%)		CLR (5.8%)	
DTV	Dative	NOM	Nominal	EXT	Extent	CLF	It-cleft	CLR	Closely-Related
LGS	Logical subj	ADV	Non-specific	LOC	Location	HLN	Headline		
PRD	Predicate		Adverbial	MNR	Manner	TTL	Title		
PUT	LOC of 'put'	BNF	Benefactive	PRP	Purpose			**Topicalization (2.6%)**	
SBJ	Subject	DIR	Direction	TMP	Temporal				
VOC	Vocative							TPC	Topic

Figure 2: Function Tags - Also shown is the percentage of each category in the Penn Treebank

emulation of the Collins parser.[3] Remarkably little modification to the parser is needed to allow it to produce function tags as part its output, without decreasing the regular Parseval metric.

The training process for the unmodified Collins parser carries out various preprocessing steps, which modify the trees in various ways before taking observations from them for the model. One of these steps is to identify and mark arguments with a parser internal tag (`-A`), using function tags as part of the heuristics for doing so. A following preprocessing step then deletes the original function tags.

We modify the Collins parser in a very simple way: the parser now retains the function tags after using them for argument identification, and so includes them in all the parameter classes. We also augment the argument identification heuristic to treat any nonterminal with any of the tags in the Syntactic group to be an argument; these are treated as synonyms for the internal tag that the parser uses to mark arguments. This therefore extends (Collins, 2003)'s use of function tags for excluding potential argument to also use them for including arguments.[4] The parser is then trained as before.

4 Function Tags: Evaluation

We compare our tagging results in isolation with the tagging systems of (Blaheta, 2003), since that work has the first highly detailed accounting of function tag results on the Penn Treebank, and with two recent tagging systems. We use both Blaheta's metric and his function tag groupings, shown in Figure 2, although our assignments are made by a fully integrated system. There are two aspects of Blaheta's metric that require discussion: First, this metric includes only constituents that were otherwise parsed correctly (ignoring function tag). Second, the metric ignores cases in which both the gold and test nonterminals are lacking function tags, since they would inflate the results.

5 Function Tags: Results

We trained the Bikel emulations of Collins' model 2 and our modified versions on sections 2-21 and tested on section 23. Scores are for all sentences, not just those with less than 40 words.

Parseval labelled recall/precision scores for the unmodified and modified parsers, show that there is almost no difference in the scores:

Parser	LR/LP
Model 2	88.12/88.31
Model 2-FuncB	88.23/88.31

We find this somewhat surprising, as we had expected that sparse data problems would arise, due to the shattering of `NP` into `NP-TMP`, `NP-SBJ`, etc.

Table 1 shows the overall results and the breakdown for the different function tag groups. For purposes of comparison, we have calculated our overall score both with and without `CLR`.[5] The (Blaheta, 2003) numbers in parentheses in Table 1 are from his feature trees specialized for the Syntactic and Semantic groups, while all his other numbers, including the overall score, are from using a single feature set for his four function tag groups.[6]

[3] Publicly available at `http://www.cis.upenn.edu/~dbikel/software.html`.

[4] Bikel's parser, in its latest version, already does something like this for Chinese and Arabic. However, the interaction with the subcat frame is different, in that it puts all nonterminals with a function tag into the miscellaneous slot in the subcat frame.

[5] (Jijkoun and de Rijke, 2004) do not state whether they are including `CLR`, but since they are comparing their results to (Blaheta and Charniak, 2000), we are assuming that they do. They do not break their results down by group.

[6] The P/R/F scores in (Blaheta, 2003)[p. 23] are internally

	— Overall —		— Breakdown by Function Tag Group —				
	w/CLR	**w/o CLR**	Syn	Sem	Top	Misc	CLR
Tag Group Frequency			*55.87%*	*36.40%*	*2.60%*	*1.03%*	*5.76%*
Model2-Ftags	88.95	90.78	95.76	84.56	93.89	17.31	65.86
Blaheta, 2003		88.28	95.16 (95.89)	79.81 (83.37)	93.72	39.44	
Jijkoun and de Rijke, 2004	88.50						
Musillo and Merlo, 2005			96.5	85.6			

Table 1: Overall Results (F-measure) and Breakdown by Function Tag Groups

Even though our tagging system results from only eliminating a few lines of code from the Collins parse, it has a higher overall score than (Blaheta, 2003), and a large increase over Blaheta's non-specialized Semantic score (79.81). It also outperforms even Blaheta's specialized Semantic score (83.37), and is very close to Blaheta's specialized score for the Syntactic group (95.89). However, since the evaluation is over a different set of non-terminals, arising from the different parsers,[7] it is difficult to draw conclusions as to which system is definitively "better". It does seem clear, though, that by integrating the function tags into the lexicalized parser, the results are roughly comparable with the post-processing work, and it is much simpler, without the need for a separate post-processing level or for specialized feature trees for the different tag groups.[8]

Our results clarify, we believe, the recent results of (Musillo and Merlo, 2005), now state-of-the-art, which extends the parser of

report a significant modification of the Henderson parser to incorporate strong notions of linguistic locality. They also manually restructure some of the function tags using tree transformations, and then train on these relabelled trees. Our results indicate that perhaps the simplest possible modification of an existing parser suffices to perform better than post-processing approaches. The linguistic sophistication of the work of (Musillo and Merlo, 2005) then provides an added boost in performance over simple integration.

6 Empty Categories: Approach

Most learning–based, phrase–structure–based (PSLB) work[9] on recovering empty categories has fallen into two classes: those which integrate empty category recovery into the parser (Dienes and Dubey, 2003a; Dienes and Dubey, 2003b) and those which recover empty categories from parser output in a post–processing step (Johnson, 2002; Levy and Manning, 2004). Levy and Manning note that thus far no PSLB post–processing approach has come close to matching the integrated approach on the most numerous types of empty categories.

However, there is a rule–based post–processing approach consisting of a set of entirely hand–designed rules (Campbell, 2004) which has better

inconsistent for the Semantic and Overall scores. We have kept the Precision and Recall and recalculated the F-measures, adjusting the Semantic score upwards from 79.15% to 79.81% and the Overall score downward from 88.63% to 88.28%.

[7] And the (Charniak, 2000) parser that (Blaheta, 2003) used has a reported F-measure of 89.5, higher than the Bikel parser used here.

[8] Our score on the Miscellaneous category is significantly lower, but as can be seen from Figure 2 and repeated in 1, this is a very rare category.

[9] As above, we consider only that work which both inputs and outputs phrase–structure trees. This notably excludes Jijkoun and de Rijke (Jijkoun and de Rijke, 2004), who have a system which seems to match the performance of Dienes and Dubey. However, they provide only aggregate statistics over all the types of empty categories, making any sort of detailed comparison impossible. Finally, it is not clear that their numbers are in fact comparable to those of Dienes and Dubey on parsed data because the metrics used are not quite equivalent, particularly for `(NP *)`s: among other differences, unlike Jijkoun and de Rijke's metric (taken from (Johnson, 2002)), Dienes and Dubey's is sensitive to the string extent of the antecedent node, penalizing them if the parser makes attachment errors involving the antecedent even if the system recovered the long–distance dependency itself correctly. Johnson noted that the two metrics did not seem to differ much for his system, but we found that evaluating our system with the laxer metric reduced error by 20% on the crucial task of restoring and finding the antecedents of `(NP *)`s, which make up almost half the empty categories in the Treebank.

results than the integrated approach. Campbell's rules make heavy use of aspects of linguistic representation unexploited by PSLB post–processing approaches, most importantly function tags and argument annotation.[10]

7 Empty Categories: Method

7.1 Runtime

The algorithm applies a series five maximum–entropy and two perceptron–based classifiers:

[1] For each PP, VP, and S node, ask the classifier NPTRACE to determine whether to insert an (NP *) as the object of a preposition, an argument of a verb, or the subject of a clause, respectively.

[2] For each node , ask NULLCOMP to determine whether or not to insert a 0 to the right.

[3] For each S node , ask WHXPINSERT to determine whether or not to insert a null *wh*–word to the left. If one should be, ask WHXPDISCERN to decide if it should be a (WHNP 0) or a (WHADVP 0).

[4] For each S which is a sister of WHNP or WHADVP, consider all possible places beneath it a *wh*–trace could be placed. Score each of them using WHTRACE, and insert a trace in the highest scoring position.

[5] For any S lacking a subject, insert (NP *).

[6] For each (NP *) in subject position, look at all NPs which c–command it. Score each of these using PROANTECEDENT, and co–index the (NP *) with the NP with the highest score. For all (NP *)s in non–subject positions, we follow Campbell in assigning the local subject as the controller.

[7] For each (NP *), ask ANTECEDENTLESS to determine whether or not to remove the co–indexing between it and its antecedent.

The sequencing of classifiers and choice of how to frame the classification decisions closely follows Campbell with the exception of finding antecedents of (NP *)s and inserting *wh*–traces, which follow Levy and Manning in using a competition–based approach. We differ from Levy and Manning in using a perceptron–based approach for these, rather than a maximum–entropy one. Also, rather than introducing an extra zero node for uncontrolled (NP *)s, we always assign a controller and then remove co–indexing from uncontrolled (NP *)s using a separate classifier.

[10]The non–PSLB system of Jijkoun and de Rijke uses function tags, and Levy and Manning mention that the lack of this information was sometimes an obstacle for them. Also, access to argument annotation inside the parser may account for a part of the good performance of Dienes and Dubey.

7.2 Training

Each of the maximum–entropy classifiers mentioned above was trained using MALLET (McCallum, 2002) over a common feature set. The most notable departure of this feature list from previous ones is in the use of function tags and argument markings, which were previously ignored for the understandable reason that though they are present in the Penn Treebank, parsers generally do not produce them. Another somewhat unusual feature examined right and left sisters.

The PROANTECEDENT perceptron classifier uses the local features of the controller and the controlled (NP *), whether the controller precedes or follows the controlled (NP *), the sequence of categories on the path between the two (with the 'turning' category marked), the length of that path, and which categories are contained anywhere along the path.

The WHTRACE perceptron classifier uses the following features each conjoined with the type of *wh*–trace being sought: the sequence of categories found on the path between the trace and its antecedent, the path length, which categories are contained anywhere along the path, the number of bounding categories crossed and whether the trace placement violates subjacency, whether or not the trace insertion site's parent is the first verb on the path, whether or not the insertion site's parent contains another verb beneath it, and if the insertion site's parent is a verb, whether or not the verb is saturated.[11]

All maximum–entropy classifiers were trained on sections 2-21 of the Penn Treebank's Wall Street Journal section; the perceptron–based classifiers were trained on sections 10-18. Section 24 was used for development testing while choosing the feature

[11]To provide the verb saturation feature, we calculated the number of times each verb in the training corpus occurs with each number of NP arguments (both overt and traces). When calculating the feature value, we compare the number of instances seen in the training corpus of the verb with the number of argument NPs it overtly has with the number of times in the corpus the verb occurs with one more argument NP.

set and other aspects of the system, and section 23 was used for the final evaluation.

8 Empty Categories: Results

8.1 Metrics

For the sake of easy comparison, we report our results using the most widely–used metric for performance on this task, that proposed by Johnson. This metric judges an entity correct if it matches the gold standard in type and string position (and, if there is an antecedent, in its label and string extent). Because Campell reports results by category using only his own metric, we use this metric to compare our results to his. There is much discussion in the literature of metrics for this task; Levy and Manning and Campbell both note that the Johnson metric fails to catch when an empty category has a correct string position but incorrect parse tree attachment. While we do not have space to discuss this issue here, the metrics they in turn propose also have significant weaknesses. In any event, we use the metrics that allow the most widespread comparison.

8.2 Comparison to Other PSLB Methods

Category	Pres	LM	J	DD
Comb. 0	**87.8**	87.0	77.1	
COMP-SBAR	**91.9**		88.0	85.5
COMP-WHNP	**61.5**		47.0	48.8
COMP-WHADVP	**69.0**			
NP *	69.1	61.1	55.6	**70.3**
Comb. *wh*–trace	**78.2**	63.3	75.2	75.3
NP *T*	80.9		80.0	**82.0**
ADVP *T*	**69.8**		56	53.6

Table 2: F1 scores comparing our system to the two PSLB post–processing systems and Dienes and Dubey's integrated system on automatically parsed trees from section 23 using Johnson's metric.

F1 scores on parsed sentences from section 23 are given in table 2. Note that our system's parsed scores were obtained using our modified version of Bikel's implementation of Collins's thesis parser which assigns function tags, while the other PSLB post–processing systems use Charniak's parser (Charniak, 2000) and Dienes and Dubey integrate empty category recovery directly into a variant of Collins's parser.

On parsed trees, our system outperforms other PSLB post–processing systems. On the most numerous category by far, `(NP *)`, our system reduces the error of the best PSLB post–processing approach by 21%. Comparing our aggregate *wh*–trace results to the others,[12] we reduce error by 41% over Levy and Manning and by 12% over Johnson.

System	Precision	Recall	F1
D&D	**78.50**	68.08	72.92
Pres	74.70	**74.62**	**74.66**

Table 3: Comparison of our system with that of Dienes and Dubey on parsed data from section 23 over the aggregation of all categories in table 2 excepting the infrequent `(WHADVP 0)`s, which they do not report but which we almost certainly outperform them on.

Performance on parsed data compared to the integrated system of Dienes and Dubey is split. We reduce error by 25% and 44% on plain `0`s and `(WHNP 0)`s, respectively and by 12% on *wh*–traces. We increase error by 4% on `(NP *)`s. Aggregating over all the categories under consideration, the more balanced precision and recall of our system puts it ahead of Dienes and Dubey's, with a 6.4% decrease in error (table 3).

8.3 Comparison to Campbell

Category	Present	Campbell
NP *	**88.8**	86.9
NP *T*	**96.3**	96.0
ADVP *T*	**82.2**	79.9
0	**99.8**	98.5

Table 4: A comparison of the present system with Campbell's rule–based system on gold–standard trees from section 23 using Campbell's metric

[12] Levy and Manning report Johnson to have an aggregate *wh*–trace score of 80, but Johnson's paper gives 80 as his score for `(NP *T*)`s only, with 56 as his score for `(ADVP *T*)`s. A similar problem seems to have occured with Levy and Manning's numbers for Dienes and Dubey on this and on `(NP *)`s. This error makes the other two systems appear to outperform Levy and Manning on *wh*–traces by a slightly larger margin than they actually do.

Classifier	Features with largest weights
NPTRACE	daughter categories, function tags, argumentness, heads, and POS tags, subjectless S...
NULLCOMP	is first daughter?, terminalness, aunt's label and POS tag, mother's head, daughters' heads, great–grandmother's label...
WHXPINSERT	is first daughter?, left sister's terminalness, labels of mother, aunt, and left sister, aunt's head...
WHXPDISCERN	words contained by grandmother, grandmother's head, aunt's head, grandmother's function tags, aunt's label, aunt's function tags...
WHTRACE	lack of subject, daughter categories, child argument information, subjacency violation, saturation, whether or not there is a verb below, path information...
PROANTECEDENT	controller's sisters' function tags, categories path contains, path length, path shape, controller's function tags, controller's sisters' heads, linear precedence information...
ANTECEDENTLESS	mother's function tags, great–grandmother's label, aunt's head ("It is difficult to..."), grandmother's function tag, mother's head...

Table 5: A few of the most highly weighted features for various classifiers

On gold-standard trees,[13] our system outperforms Campbell's rule–based system on all four categories, reducing error by 87% on `0`s,[14] by 11% on `(ADVP *T*)`s, by 7% on `(NP *T*)`s, and by 8% on the extremely numerous `(NP *)`s.

9 Empty Categories: Discussion

We have shown that a PSLB post–processing approach can outperform the state–of–the–art integrated approach of Dienes and Dubey.[15] Given that their modifications to Collins's parser caused a decrease in local phrase structure parsing accuracy due to sparse data difficulties (Dienes and Dubey, 2003a), our post–processing approach seems to be an especially attractive choice. We have further shown that our PSLB approach, using only simple, unconjoined features, outperforms Campbell's state–of–the–art, complex system on gold–standard data, suggesting that much of the power of his system lies in his richer linguistic representation and his structuring of decisions rather than the hand–designed rules.

We have also compared our system to that of Levy and Manning which is based on a similar learning technique and have shown large increases in performance on all of the most common types of empty categories; this increase seems to have come almost entirely from an enrichment of the linguistic representation and a slightly different structuring of the problem, rather than any use of more powerful machine–learning techniques

We speculate that the primary source of our performance increase is the enrichment of the linguistic representation with function tags and argument markings from the parser's first stage, as table 5 attests. We also note that several classifiers make use of the properties of aunt nodes, which have previously been exploited only in a limite form in Johnson's patterns. For example, ANTECEDENTLESS uses the aunt's head word to learn an entire class of uncontrolled PRO constructions like "It is difficult `(NP *)` to imagine living on Mars."

10 Conclusion

This work has presented a two stage parser that recovers Penn Treebank style syntactic analyses of new sentences including skeletal syntactic structure, and, for the first time, both function tags and empty categories. The accuracy of the first-stage parser on the standard Parseval metric matches that of the (Collins, 2003) parser on which it is based, despite the data fragmentation caused by the greatly enriched space of possible node labels for the Collins statistical model. This first stage simultaneously achieves near state-of-the-art performance on recovering function tags with minimal modifications to the underlying parser, modifying less than ten lines

[13] Only aggregate statistics over a different set of empty categories were available for Campbell on parsed data, making a comparison impossible.

[14] Note that for comparison with Campbell, the `0` numbers here exclude `(WHNP 0)`s and `(WHADVP 0)`s.

[15] And therefore also very likely outperforms the dependency–based post–processing approach of Jijkoun and de Rijke, even if its performance does in fact equal Dienes and Dubey's.

of code. We speculate that this success is due to the lexicalization of the Collins model, combined with the sophisticated backoff structure already built into the Collins model. The second stage achieves state-of-the-art performance on the recovery of empty categories by combining the linguistically-informed architecture of (Campbell, 2004) and a rich feature set with the power of modern machine learning methods. This work provides an example of how small enrichments in linguistic representation and changes in the structure of the problem having significant effects on the performance of a machine–learning–based system. More concretely, we showed for the first time that a PSLB post–processing system can outperform the state–of–the–art for both rule–based post–processing and integrated approaches to the empty category restoration problem.

Most importantly from the point of view of the authors, we have constructed a system that recovers sufficiently rich syntactic structure based on the Penn Treebank to provide rich syntactic guidance for the recovery of predicate-argument structure in the near future. We also expect that productivity of syntactic annotation of further genres of English will be significantly enhanced by the use of this new tool, and hope to have practical evidence of this in the near future.

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.

Daniel M. Bikel. 2004. *On the Parameter Space of Lexicalized Statistical Parsing Models*. Ph.D. thesis, Department of Computer and Information Sciences, University of Pennsylvania.

E. Black, S. Abney, D. Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the Fourth DARPA Workshop on Speech and Natural Language*, pages 306–311, CA.

Don Blaheta and Eugene Charniak. 2000. Assigning function tags to parsed text. In *Proceedings of the 1st Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 234–240, Seattle.

Don Blaheta. 2003. *Function Tagging*. Ph.D. thesis, Brown University.

Richard Campbell. 2004. Using linguistic principles to recover empty categories. In *Proceedings of ACL*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Annual Meeting of the North American Chapter of the Association for Computational Linguistics*.

Michael Collins. 1997. Three generative, lexicalized models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, Madrid.

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

Peter Dienes and Amit Dubey. 2003a. Antecedent recovery: Experiments with a trace tagger. In *Proceedings of EMNLP*.

Peter Dienes and Amit Dubey. 2003b. Deep processing by combining shallow methods. In *Proceedings of ACL*.

James Henderson. 2003. Inducing history representations for broad coverage statistical parsing. In *Proceedings of NLT-NAACL 2003*, Edmonton, Alberta, Canada. Association for Computational Linguistics.

Valentin Jijkoun and Maarten de Rijke. 2004. Enriching the output of a parser using memory-based learning. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain.

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40st Annual Meeting of the Association for Computational Linguistics*, Philadelphia, PA.

Roger Levy and Christopher Manning. 2004. Deep dependencies from context–free statistical parsers: correcting the surface dependency approximation. In *Proceedings of the ACL*.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Gabriele Musillo and Paolo Merlo. 2005. Lexical and structural biases for function parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 83–92, Vancouver, British Columbia, October. Association for Computational Linguistics.

Exploiting Semantic Role Labeling, WordNet and Wikipedia for Coreference Resolution

Simone Paolo Ponzetto and **Michael Strube**
EML Research gGmbH
Schloss-Wolfsbrunnenweg 33
69118 Heidelberg, Germany
http://www.eml-research.de/nlp

Abstract

In this paper we present an extension of a machine learning based coreference resolution system which uses features induced from different semantic knowledge sources. These features represent knowledge mined from WordNet and Wikipedia, as well as information about semantic role labels. We show that semantic features indeed improve the performance on different referring expression types such as pronouns and common nouns.

1 Introduction

The last years have seen a boost of work devoted to the development of machine learning based coreference resolution systems (Soon et al., 2001; Ng & Cardie, 2002; Yang et al., 2003; Luo et al., 2004, inter alia). While machine learning has proved to yield performance rates fully competitive with rule based systems, current coreference resolution systems are mostly relying on rather shallow features, such as the distance between the coreferent expressions, string matching, and linguistic form. However, the literature emphasizes since the very beginning the relevance of world knowledge and inference for coreference resolution (Charniak, 1973).

This paper explores whether coreference resolution can benefit from semantic knowledge sources. More specifically, whether a machine learning based approach to coreference resolution can be improved and *which phenomena* are affected by such information. We investigate the use of the WordNet and Wikipedia taxonomies for extracting *semantic similarity* and *relatedness* measures, as well as semantic parsing information in terms of *semantic role labeling* (Gildea & Jurafsky, 2002, SRL henceforth).

We believe that the lack of semantics in the current systems leads to a performance bottleneck. In order to correctly identify the discourse entities which are referred to in a text, it seems essential to reason over the lexical semantic relations, as well as the event representations embedded in the text. As an example, consider a fragment from the Automatic Content Extraction (ACE) 2003 data.

(1) But frequent visitors say that given the sheer weight of **the country**'s totalitarian ideology and generations of mass indoctrination, changing **this country**'s course will be something akin to turning a huge ship at sea. Opening **North Korea** up, even modestly, and exposing **people** to the idea that Westerners – and South Koreans – are not devils, alone represents an extraordinary change. [...] as **his people** begin to get a clearer idea of the deprivation **they** have suffered, especially relative to **their** neighbors. "**This** is **a society** that has been focused most of all on stability, [...]".

In order to correctly resolve the anaphoric expressions highlighted in bold, it seems that some kind of lexical semantic and encyclopedic knowledge is required. This includes that *North Korea* is a *country*, that *countries* consist of *people* and are *societies*. The resolution requires an encyclopedia (i.e. Wikipedia) look-up and reasoning on the content relatedness holding between the different expressions (i.e. as a path measure along the links of the WordNet and Wikipedia taxonomies). Event representations seem also to be important for coreference resolution, as shown below:

(2) A state commission of inquiry into the sinking of the Kursk will convene in Moscow on Wednesday, **the Interfax news agency** reported. **It** said that the diving operation will be completed by the end of next week.

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 192–199,
New York, June 2006.

In this example, knowing that *the Interfax news agency* is the AGENT of the *report* predicate and *It* being the AGENT of *say* could trigger the (semantic parallelism based) inference required to correctly link the two expressions, in contrast to anchoring the pronoun to *Moscow*. SRL provides the semantic relationships that constituents have with predicates, thus allowing us to include such document-level *event descriptive information* into the relations holding between referring expressions (REs).

Instead of exploring different kinds of data representations, task definitions or machine learning techniques (Ng & Cardie, 2002; Yang et al., 2003; Luo et al., 2004) we focus on a few promising semantic features which we evaluate in a controlled environment. That way we try to overcome the plateauing in performance in coreference resolution observed by Kehler et al. (2004).

2 Related Work

Vieira & Poesio (2000), Harabagiu et al. (2001), and Markert & Nissim (2005) explore the use of WordNet for different coreference resolution subtasks, such as resolving bridging reference, *other*- and definite NP anaphora, and MUC-style coreference resolution. All of them present systems which infer coreference relations from a set of potential antecedents by means of a WordNet search. Our approach to WordNet here is to cast the search results in terms of semantic similarity measures. Their output can be used as features for a learner. These measures are not specifically developed for coreference resolution but simply taken 'off-the-shelf' and applied to our task without any specific tuning — i.e. in contrast to Harabagiu et al. (2001), who weight WordNet relations differently in order to compute the confidence measure of the path.

To the best of our knowledge, we do not know of any previous work using Wikipedia or SRL for coreference resolution. In the case of SRL, this layer of semantic context abstracts from the specific lexical expressions used, and therefore represents a higher level of abstraction than (still related) work involving predicate argument statistics. Kehler et al. (2004) observe no significant improvement due to predicate argument statistics. The improvement reported by Yang et al. (2005) is rather caused by their twin-candidate model than by the semantic knowledge. Employing SRL is closer in spirit to Ji et al. (2005), who explore the employment of the ACE 2004 relation ontology as a semantic filter.

3 Coreference Resolution Using Semantic Knowledge Sources

3.1 Corpora Used

To establish a competitive coreference resolver, the system was initially prototyped using the MUC-6 and MUC-7 data sets (Chinchor & Sundheim, 2003; Chinchor, 2001), using the standard partitioning of 30 texts for training and 20-30 texts for testing. Then, we moved on and developed and tested the system with the ACE 2003 Training Data corpus (Mitchell et al., 2003)[1]. Both the Newswire (NWIRE) and Broadcast News (BNEWS) sections where split into 60-20-20% document-based partitions for training, development, and testing, and later per-partition merged (MERGED) for system evaluation. The distribution of coreference chains and referring expressions is given in Table 1.

3.2 Learning Algorithm

For learning coreference decisions, we used a Maximum Entropy (Berger et al., 1996) model. This was implemented using the MALLET library (McCallum, 2002). To prevent the model from overfitting, we employed a tunable Gaussian prior as a smoothing method. The best parameter value is found by searching in the [0,10] interval with step value of 0.5 for the variance parameter yielding the highest MUC score F-measure on the development data.

Coreference resolution is viewed as a binary classification task: given a pair of REs, the classifier has to decide whether they are coreferent or not. The MaxEnt model produces a probability for each category y (coreferent or not) of a candidate pair, conditioned on the context x in which the candidate occurs. The conditional probability is calculated by:

$$p(y|x) = \frac{1}{Z_x}\left[\sum_i \lambda_i f_i(x,y)\right]$$

[1] We used the training data corpus only, as the availability of the test data is restricted to ACE participants. Therefore, the results we report cannot be compared directly with those using the official test data.

	BNEWS (147 docs – 33,479 tokens)				NWIRE (105 docs – 57,205 tokens)			
	#coref ch.	#pron.	#comm. nouns	#prop. names	#coref ch.	#pron.	#comm. nouns	#prop. names
TRAIN.	587	876	572	980	904	1,037	1,210	2,023
DEVEL	201	315	163	465	399	358	485	923
TEST	228	291	238	420	354	329	484	712
TOTAL	1,016	1,482	973	1,865	1,657	1,724	2,179	3,658
TOTAL (%)		34.3%	22.5%	43.2%		22.8%	28.8%	48.4%

Table 1: Partitions of the ACE 2003 training data corpus

where $f_i(x, y)$ is the value of feature i on outcome y in context x, and λ_i is the weight associated with i in the model. Z_x is a normalization constant. The features used in our model are all binary-valued feature functions (or indicator functions), e.g.

$$f_{\text{I\_SEMROLE}}(\text{ARG0/RUN, COREF}) = \begin{cases} 1 & \text{if candidate pair is coreferent and antecedent is the semantic argument ARG0 of predicate } \textit{run} \\ 0 & \text{else} \end{cases}$$

In our system, a set of pre-processing components including a POS tagger (Giménez & Màrquez, 2004), NP chunker (Kudoh & Matsumoto, 2000) and the *Alias-I LingPipe* Named Entity Recognizer[2] is applied to the text in order to identify the noun phrases, which are further taken as referring expressions (REs) to be used for instance generation. Therefore, we use automatically extracted noun phrases, rather than assuming perfect NP chunking. This is in contrast to other related works in coreference resolution (e.g. Luo et al. (2004), Kehler et al. (2004)).

Instances are created following Soon et al. (2001). We create a positive training instance from each pair of adjacent coreferent REs. Negative instances are obtained by pairing the anaphoric REs with any RE occurring between the anaphor and the antecedent. During testing each text is processed from left to right: each RE is paired with any preceding RE from right to left, until a pair labeled as coreferent is output, or the beginning of the document is reached. The classifier imposes a partitioning on the available REs by clustering each set of expressions labeled as coreferent into the same coreference chain.

[2] `http://alias-i.com/lingpipe`

3.3 Baseline System Features

Following Ng & Cardie (2002), our baseline system reimplements the Soon et al. (2001) system. The system uses 12 features. Given a potential antecedent RE_i and a potential anaphor RE_j the features are computed as follows[3].

(a) Lexical features

STRING_MATCH T if RE_i and RE_j have the same spelling, else F.

ALIAS T if one RE is an alias of the other; else F.

(b) Grammatical features

I_PRONOUN T if RE_i is a pronoun; else F.

J_PRONOUN T if RE_j is a pronoun; else F.

J_DEF T if RE_j starts with *the*; else F.

J_DEM T if RE_j starts with *this*, *that*, *these*, or *those*; else F.

NUMBER T if both RE_i and RE_j agree in number; else F.

GENDER U if either RE_i or RE_j have an undefined gender. Else if they are both defined and agree T; else F.

PROPER_NAME T if both RE_i and RE_j are proper names; else F.

APPOSITIVE T if RE_j is in apposition with RE_i; else F.

(c) Semantic features

WN_CLASS U if either RE_i or RE_j have an undefined WordNet semantic class. Else if they both have a defined one and it is the same T; else F.

(d) Distance features

DISTANCE how many sentences RE_i and RE_j are apart.

[3] Possible values are U(nknown), T(rue) and F(alse). Note that in contrast to Ng & Cardie (2002) we interpret ALIAS as a lexical feature, as it solely relies on string comparison and acronym string matching.

3.4 WordNet Features

In the baseline system semantic information is limited to WordNet semantic class matching. Unfortunately, a WordNet semantic class lookup exhibits problems such as coverage, sense proliferation and ambiguity[4], which make the WN_CLASS feature very noisy. We enrich the semantic information available to the classifier by using semantic similarity measures based on the WordNet taxonomy (Pedersen et al., 2004). The measures we use include path length based measures (Rada et al., 1989; Wu & Palmer, 1994; Leacock & Chodorow, 1998), as well as ones based on information content (Resnik, 1995; Jiang & Conrath, 1997; Lin, 1998).

In our case, the measures are obtained by computing the similarity scores between the head lemmata of each potential antecedent-anaphor pair. In order to overcome the sense disambiguation problem, we factorise over all possible sense pairs: given a candidate pair, we take the cross product of each antecedent and anaphor sense to form pairs of synsets. For each measure WN_SIMILARITY, we compute the similarity score for all synset pairs, and create the following features.

WN_SIMILARITY_BEST the *highest* similarity score from all $\langle \text{SENSE}_{RE_i,n}, \text{SENSE}_{RE_j,m} \rangle$ synset pairs.

WN_SIMILARITY_AVG the *average* similarity score from all $\langle \text{SENSE}_{RE_i,n}, \text{SENSE}_{RE_j,m} \rangle$ synset pairs.

Pairs containing REs which cannot be mapped to WordNet synsets are assumed to have a null similarity measure.

3.5 Wikipedia Features

Wikipedia is a multilingual Web-based free-content encyclopedia[5]. The English version, as of 14 February 2006, contains 971,518 articles with 16.8 million internal hyperlinks thus providing a large coverage available knowledge resource. In addition, since May 2004 it provides also a taxonomy by means of the *category feature*: articles can be placed in one or more categories, which are further categorized to provide a category tree. In practice, the taxonomy is not designed as a strict hierarchy or tree of categories, but allows multiple categorisation schemes to co-exist simultaneously. Because each article can appear in more than one category, and each category can appear in more than one parent category, the categories do not form a tree structure, but a more general directed graph. As of December 2005, 78% of the articles have been categorized into 87,000 different categories.

Wikipedia mining works as follows (for an in-depth description of the methods for computing semantic relatedness in Wikipedia see Strube & Ponzetto (2006)): given the candidate referring expressions RE_i and RE_j we first pull the pages they refer to. This is accomplished by querying the page titled as the head lemma or, in the case of NEs, the full NP. We follow all redirects and check for disambiguation pages, i.e. pages for ambiguous entries which contain links only (e.g. *Lincoln*). If a disambiguation page is hit, we first get all the hyperlinks in the page. If a link containing the other queried RE is found (i.e. a link containing *president* in the *Lincoln* page), the linked page (*President of the United States*) is returned, else we return the first article linked in the disambiguation page. Given a candidate coreference pair $\text{RE}_{i/j}$ and the Wikipedia pages $P_{RE_{i/j}}$ they point to, obtained by querying pages titled as $T_{RE_{i/j}}$, we extract the following features:

I/J_GLOSS_CONTAINS U if no Wikipedia page titled $T_{RE_{i/j}}$ is available. Else T if the first paragraph of text of $P_{RE_{i/j}}$ contains $T_{RE_{j/i}}$; else F.

I/J_RELATED_CONTAINS U if no Wikipedia page titled as $T_{RE_{i/j}}$ is available. Else T if at least one Wikipedia hyperlink of $P_{RE_{i/j}}$ contains $T_{RE_{j/i}}$; else F.

I/J_CATEGORIES_CONTAINS U if no Wikipedia page titled as $T_{RE_{i/j}}$ is available. Else T if the list of categories $P_{RE_{i/j}}$ belongs to contains $T_{RE_{j/i}}$; else F.

GLOSS_OVERLAP the overlap score between the first paragraph of text of P_{RE_i} and P_{RE_j}. Following Banerjee & Pedersen (2003) we compute the score as $\sum_n m^2$ for n phrasal m-word overlaps.

[4] Following the system to be replicated, we simply mapped each RE to the first WordNet sense of the head noun.

[5] Wikipedia can be downloaded at `http://download.wikimedia.org/`. In our experiments we use the English Wikipedia database dump from 19 February 2006.

Additionally, we use the Wikipedia category graph. We ported the WordNet similarity path length based measures to the Wikipedia category graph. However, the category relations in Wikipedia cannot only be interpreted as corresponding to *is-a* links in a taxonomy since they denote meronymic relations as well. Therefore, the Wikipedia-based measures are to be taken as semantic relatedness measures. The measures from Rada et al. (1989), Leacock & Chodorow (1998) and Wu & Palmer (1994) are computed in the same way as for WordNet. Path search takes place as a depth-limited search of maximum depth of 4 for a least common subsumer. We noticed that limiting the search improves the results as it yields a better correlation of the relatedness scores with human judgements (Strube & Ponzetto, 2006). This is due to the high regions of the Wikipedia category tree being too strongly connected.

In addition, we use the measure from Resnik (1995), which is computed using an intrinsic information content measure relying on the hierarchical structure of the category tree (Seco et al., 2004). Given $P_{RE_{i/j}}$ and the lists of categories $C_{RE_{i/j}}$ they belong to, we factorise over all possible category pairs. That is, we take the cross product of each antecedent and anaphor category to form pairs of 'Wikipedia synsets'. For each measure WIKI_RELATEDNESS, we compute the relatedness score for all category pairs, and create the following features.

WIKI_RELATEDNESS_BEST the *highest* relatedness score from all $\langle C_{RE_i,n}, C_{RE_j,m} \rangle$ category pairs.

WIKI_RELATEDNESS_AVG the *average* relatedness score from all $\langle C_{RE_i,n}, C_{RE_j,m} \rangle$ category pairs.

3.6 Semantic Role Features

The last semantic knowledge enhancement for the baseline system uses SRL information. In our experiments we use the ASSERT parser (Pradhan et al., 2004), an SVM based semantic role tagger which uses a full syntactic analysis to automatically identify all verb predicates in a sentence together with their semantic arguments, which are output as PropBank arguments (Palmer et al., 2005). It is often the case that the semantic arguments output by the parser do not align with any of the previously identified noun phrases. In this case, we pass a semantic role label to a RE only when the two phrases share the same head. Labels have the form "$\text{ARG}_1\text{\_pred}_1 \ldots \text{ARG}_n\text{\_pred}_n$" for n semantic roles filled by a constituent, where each semantic argument label is always defined with respect to a predicate. Given such level of semantic information available at the RE level, we introduce two new features[6].

I_SEMROLE the semantic role argument-predicate pairs of RE_i.

J_SEMROLE the semantic role argument-predicate pairs of RE_j.

For the ACE 2003 data, 11,406 of 32,502 automatically extracted noun phrases were tagged with 2,801 different argument-predicate pairs.

4 Experiments

4.1 Performance Metrics

We report in the following tables the MUC score (Vilain et al., 1995). Scores in Table 2 are computed for all noun phrases appearing in either the key or the system response, whereas Tables 3 and 4 refer to scoring only those phrases which appear in both the key and the response. We therefore discard those responses not present in the key, as we are interested in establishing the upper limit of the improvements given by our semantic features. That is, we want to define a baseline against which to establish the contribution of the semantic information sources explored here for coreference resolution.

In addition, we report the accuracy score for all three types of ACE mentions, namely pronouns, common nouns and proper names. Accuracy is the percentage of REs of a given mention type correctly resolved divided by the total number of REs of the same type given in the key. A RE is said to be correctly resolved when both it and its direct antecedent are placed by the key in the same coreference class.

[6] During prototyping we experimented unpairing the arguments from the predicates, which yielded worse results. This is supported by the PropBank arguments always being defined with respect to a target predicate. Binarizing the features — i.e. do RE_i and RE_j have the same argument or predicate label with respect to their closest predicate? — also gave worse results.

	MUC-6			MUC-7		
original Soon et al.	R 58.6	P 67.3	F_1 62.3	R 56.1	P 65.5	F_1 60.4
duplicated baseline	64.9	65.6	65.3	55.1	68.5	61.1

Table 2: Results on MUC

4.2 Feature Selection

For determining the relevant feature sets we follow an iterative procedure similar to the wrapper approach for feature selection (Kohavi & John, 1997) using the development data. The feature subset selection algorithm performs a hill-climbing search along the feature space. We start with a model based on all available features. Then we train models obtained by removing one feature at a time. We choose the worst performing feature, namely the one whose removal gives the largest improvement based on the MUC score F-measure, and remove it from the model. We then train classifiers removing each of the remaining features separately from the enhanced model. The process is iteratively run as long as significant improvement is observed.

4.3 Results

Table 2 compares the results between our duplicated Soon baseline and the original system. We assume that the slight improvements of our system are due to the use of current pre-processing components and another classifier. Tables 3 and 4 show a comparison of the performance between our baseline system and the ones incremented with semantic features. Performance improvements are highlighted in bold[7].

4.4 Discussion

The tables show that *semantic features improve system recall*, rather than acting as a 'semantic filter' improving precision. Semantics therefore seems to trigger a response in cases where more shallow features do not seem to suffice (see examples (1-2)).

Different feature sources account for different RE type improvements. WordNet and Wikipedia features tend to increase performance on common nouns, whereas SRL improves pronouns. WordNet features are able to improve by 14.3% and 7.7% the accuracy rate for common nouns on the BNEWS and NWIRE datasets (+34 and +37 correctly resolved common nouns out of 238 and 484 respectively), whereas employing Wikipedia yields slightly smaller improvements (+13.0% and +6.6% accuracy increase on the same datasets). Similarly, when SRL features are added to the baseline system, we register an increase in the accuracy rate for pronouns, ranging from 0.7% in BNEWS and NWIRE up to 4.2% in the MERGED dataset (+26 correctly resolved pronouns out of 620).

If semantics helps for pronouns and common nouns, it does not affect performance on proper names, where features such as string matching and alias suffice. This suggests that semantics plays a role in pronoun and common noun resolution, where surface features cannot account for complex preferences and semantic knowledge is required.

The best accuracy improvement on pronoun resolution is obtained on the MERGED dataset. This is due to making more data available to the classifier, as the SRL features are very sparse and inherently suffer from data fragmentation. Using a larger dataset highlights the importance of SRL, whose features are never removed in any feature selection process[8]. The accuracy on common nouns shows that features induced from Wikipedia are competitive with the ones from WordNet. The performance gap on all three datasets is quite small, which indicates the usefulness of using an encyclopedic knowledge base as a replacement for a lexical taxonomy.

As a consequence of having different knowledge sources accounting for the resolution of different RE types, the best results are obtained by (1) *combining features* generated *from different sources*; (2) *performing feature selection*. When combining different feature sources, we register an accuracy improvement on pronouns and common nouns, as well as an increase in F-measure due to a higher recall.

Feature selection always improves results. This is due to the fact that our full feature set is ex-

[7] All changes in F-measure are statistically significant at the 0.05 level or higher. We follow Soon et al. (2001) in performing a simple one-tailed, paired sample t-test between the baseline system's MUC score F-measure and each of the other systems' F-measure scores on the test documents.

[8] To our knowledge, most of the recent work in coreference resolution on the ACE data keeps the document source separated for evaluation. However, we believe that document source independent evaluation provides useful insights on the robustness of the system (cf. the CoNLL 2005 shared task cross-corpora evaluation).

	BNEWS						NWIRE					
	R	P	F_1	A_p	A_{cn}	A_{pn}	R	P	F_1	A_p	A_{cn}	A_{pn}
baseline	46.7	86.2	60.6	36.4	10.5	44.0	56.7	88.2	69.0	37.6	23.1	55.6
+WordNet	**54.8**	86.1	**66.9**	**36.8**	**24.8**	**47.6**	**61.3**	84.9	**71.2**	**38.9**	**30.8**	55.5
+Wiki	**52.7**	**86.8**	**65.6**	36.1	**23.5**	**46.2**	**60.6**	83.6	**70.3**	**38.0**	**29.7**	55.2
+SRL	**53.3**	85.1	**65.5**	**37.1**	**13.9**	**46.2**	**58.0**	**89.0**	**70.2**	**38.3**	**25.0**	**56.0**
all features	**59.1**	84.4	**69.5**	**37.5**	**27.3**	**48.1**	**63.1**	83.0	**71.7**	**39.8**	**31.8**	52.8

Table 3: Results on the ACE 2003 data (BNEWS and NWIRE sections)

	R	P	F_1	A_p	A_{cn}	A_{pn}
baseline	54.5	88.0	67.3	34.7	20.4	53.1
+WordNet	**56.7**	87.1	**68.6**	**35.6**	**28.5**	49.6
+Wikipedia	**55.8**	87.5	**68.1**	**34.8**	**26.0**	50.5
+SRL	**56.3**	**88.4**	**68.8**	**38.9**	**21.6**	51.7
all features	**61.0**	84.2	**70.7**	**38.9**	**29.9**	51.2

Table 4: Results ACE (merged BNEWS/NWIRE)

Feature set	F_1
baseline (Soon w/o DISTANCE)	58.4%
+WIKI_WU_PALMER_BEST	+4.3%
+J_SEMROLE	+1.8%
+WIKI_PATH_AVG	+1.2%
+I_SEMROLE	+0.8%
+WN_WU_PALMER_BEST	+0.7%

Table 5: Feature selection (BNEWS section)

tremely redundant: in order to explore the usefulness of the knowledge sources we included overlapping features (i.e. using *best* and *average* similarity/relatedness measures at the same time), as well as features capturing the same phenomenon from different point of views (i.e. using *multiple* measures at the same time). In order to yield the desired performance improvements, it turns out to be essential to filter out irrelevant features.

Table 5 shows the relevance of the best performing features on the BNEWS section. As our feature selection mechanism chooses the best set of features by removing them (see Section 4.2), we evaluate the contributions of the remaining features as follows. We start with a baseline system using all the features from Soon et al. (2001) that were not removed in the feature selection process (i.e. DISTANCE). We then train classifiers combining the current feature set with each feature in turn. We then choose the best performing feature based on the MUC score F-measure and add it to the model. We iterate the process until all features are added to the baseline system. The table indicates that all knowledge sources are relevant for coreference resolution, as it includes SRL, WordNet and Wikipedia features. The Wikipedia features rank high, indicating again that it provides a valid knowledge base.

5 Conclusions and Future Work

The results are somehow surprising, as one would not expect a community-generated categorization to be almost as informative as a well structured lexical taxonomy such as WordNet. Nevertheless Wikipedia offers promising results, which we expect to improve as well as the encyclopedia goes under further development.

In this paper we investigated the effects of using different semantic knowledge sources within a machine learning based coreference resolution system. This involved mining the WordNet taxonomy and the Wikipedia encyclopedic knowledge base, as well as including semantic parsing information, in order to induce semantic features for coreference learning. Empirical results show that coreference resolution benefits from semantics. The generated model is able to learn selectional preferences in cases where surface morpho-syntactic features do not suffice, i.e. pronoun and common name resolution. While the results given by using 'the free encyclopedia that anyone can edit' are satisfactory, major improvements can come from developing efficient query strategies – i.e. a more refined disambiguation technique taking advantage of the context in which the queries (e.g. referring expressions) occur.

Future work will include turning Wikipedia into an ontology with well defined taxonomic relations, as well as exploring its usefulness of for other NLP applications. We believe that an interesting aspect of Wikipedia is that it offers large coverage resources for many languages, thus making it a natural choice for multilingual NLP systems.

Semantics plays indeed a role in coreference resolution. But semantic features are expensive to

compute and the development of efficient methods is required to embed them into large scale systems. Nevertheless, we believe that exploiting semantic knowledge in the manner we described will assist the research on coreference resolution to overcome the plateauing in performance observed by Kehler et al. (2004).

Acknowledgements: This work has been funded by the Klaus Tschira Foundation, Heidelberg, Germany. The first author has been supported by a KTF grant (09.003.2004). We thank Katja Filippova, Margot Mieskes and the three anonymous reviewers for their useful comments.

References

Banerjee, S. & T. Pedersen (2003). Extended gloss overlap as a measure of semantic relatedness. In *Proc. of IJCAI-03*, pp. 805–810.

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

Charniak, E. (1973). Jack and Janet in search of a theory of knowledge. In *Advance Papers from the Third International Joint Conference on Artificial Intelligence, Stanford, Cal.*, pp. 337–343.

Chinchor, N. (2001). *Message Understanding Conference (MUC) 7*. LDC2001T02, Philadelphia, Penn: Linguistic Data Consortium.

Chinchor, N. & B. Sundheim (2003). *Message Understanding Conference (MUC) 6*. LDC2003T13, Philadelphia, Penn: Linguistic Data Consortium.

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

Giménez, J. & L. Màrquez (2004). SVMTool: A general POS tagger generator based on support vector machines. In *Proc. of LREC '04*, pp. 43–46.

Harabagiu, S. M., R. C. Bunescu & S. J. Maiorano (2001). Text and knowledge mining for coreference resolution. In *Proc. of NAACL-01*, pp. 55–62.

Ji, H., D. Westbrook & R. Grishman (2005). Using semantic relations to refine coreference decisions. In *Proc. HLT-EMNLP '05*, pp. 17–24.

Jiang, J. J. & D. W. Conrath (1997). Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the 10th International Conference on Research in Computational Linguistics (ROCLING)*.

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.

Kohavi, R. & G. H. John (1997). Wrappers for feature subset selection. *Artificial Intelligence Journal*, 97(1-2):273–324.

Kudoh, T. & Y. Matsumoto (2000). Use of Support Vector Machines for chunk identification. In *Proc. of CoNLL-00*, pp. 142–144.

Leacock, C. & M. Chodorow (1998). Combining local context and WordNet similarity for word sense identification. In C. Fellbaum (Ed.), *WordNet. An Electronic Lexical Database*, Chp. 11, pp. 265–283. Cambridge, Mass.: MIT Press.

Lin, D. (1998). An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, pp. 296–304.

Luo, X., A. Ittycheriah, H. Jing, N. Kambhatla & S. Roukos (2004). A mention-synchronous coreference resolution algorithm based on the Bell Tree. In *Proc. of ACL-04*, pp. 136–143.

Markert, K. & M. Nissim (2005). Comparing knowledge sources for nominal anaphora resolution. *Computational Linguistics*, 31(3):367–401.

McCallum, A. K. (2002). *MALLET: A Machine Learning for Language Toolkit*.

Mitchell, A., S. Strassel, M. Przybocki, J. Davis, G. Doddington, R. Grishman, A. Meyers, A. Brunstain, L. Ferro & B. Sundheim (2003). *TIDES Extraction (ACE) 2003 Multilingual Training Data*. LDC2004T09, Philadelphia, Penn.: Linguistic Data Consortium.

Ng, V. & C. Cardie (2002). Improving machine learning approaches to coreference resolution. In *Proc. of ACL-02*, pp. 104–111.

Palmer, M., D. Gildea & P. Kingsbury (2005). The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–105.

Pedersen, T., S. Patwardhan & J. Michelizzi (2004). WordNet::Similarity – Measuring the relatedness of concepts. In *Companion Volume of the Proceedings of the Human Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 267–270.

Pradhan, S., W. Ward, K. Hacioglu, J. H. Martin & D. Jurafsky (2004). Shallow semantic parsing using Support Vector Machines. In *Proc. of HLT-NAACL-04*, pp. 233–240.

Rada, R., H. Mili, E. Bicknell & M. Blettner (1989). Development and application of a metric to semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, 19(1):17–30.

Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. In *Proc. of IJCAI-95*, Vol. 1, pp. 448–453.

Seco, N., T. Veale & J. Hayes (2004). An intrinsic information content metric for semantic similarity in WordNet. In *Proc. of ECAI-04*, pp. 1089–1090.

Soon, W. M., H. T. Ng & D. C. Y. Lim (2001). A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Strube, M. & S. P. Ponzetto (2006). WikiRelate! Computing semantic relatedness using Wikipedia. In *Proc. of AAAI-06*.

Vieira, R. & M. Poesio (2000). An empirically-based system for processing definite descriptions. *Computational Linguistics*, 26(4):539–593.

Vilain, M., J. Burger, J. Aberdeen, D. Connolly & L. Hirschman (1995). A model-theoretic coreference scoring scheme. In *Proceedings of the 6th Message Understanding Conference (MUC-6)*, pp. 45–52.

Wu, Z. & M. Palmer (1994). Verb semantics and lexical selection. In *Proc. of ACL-94*, pp. 133–138.

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.

Yang, X., G. Zhou, J. Su & C. L. Tan (2003). Coreference resolution using competition learning approach. In *Proc. of ACL-03*, pp. 176–183.

Identifying and Analyzing Judgment Opinions

Soo-Min Kim and Eduard Hovy
USC Information Sciences Institute
4676 Admiralty Way, Marina del Rey, CA 90292
{skim, hovy}@ISI.EDU

Abstract

In this paper, we introduce a methodology for analyzing judgment opinions. We define a judgment opinion as consisting of a valence, a holder, and a topic. We decompose the task of opinion analysis into four parts: 1) recognizing the opinion; 2) identifying the valence; 3) identifying the holder; and 4) identifying the topic. In this paper, we address the first three parts and evaluate our methodology using both intrinsic and extrinsic measures.

1 Introduction

Recently, many researchers and companies have explored the area of opinion detection and analysis. With the increased immersion of Internet users has come a proliferation of opinions available on the web. Not only do we read more opinions from the web, such as in daily news editorials, but also we post more opinions through mechanisms such as governmental web sites, product review sites, news group message boards and personal blogs. This phenomenon has opened the door for massive opinion collection, which has potential impact on various applications such as public opinion monitoring and product review summary systems.

Although in its infancy, many researchers have worked in various facets of opinion analysis. Pang et al. (2002) and Turney (2002) classified sentiment polarity of reviews at the document level. Wiebe et al. (1999) classified sentence level subjectivity using syntactic classes such as adjectives, pronouns and modal verbs as features. Riloff and Wiebe (2003) extracted subjective expressions from sentences using a bootstrapping pattern learning process. Yu and Hatzivassiloglou (2003) identified the polarity of opinion sentences using semantically oriented words. These techniques were applied and examined in different domains, such as customer reviews (Hu and Liu 2004) and news articles[1]. These researchers use lists of opinion-bearing clue words and phrases, and then apply various additional techniques and refinements.

Along with many opinion researchers, we participated in a large pilot study, sponsored by NIST, which concluded that it is very difficult to define what an opinion is in general. Moreover, an expression that is considered as an opinion in one domain might not be an opinion in another. For example, the statement "The screen is very big" might be a positive review for a wide screen desktop review, but it could be a mere fact in general newspaper text. This implies that it is hard to apply opinion bearing words collected from one domain to an application for another domain. One might therefore need to collect opinion clues within individual domains. In case we cannot simply find training data from existing sources, such as news article analysis, we need to manually annotate data first.

Most opinions are of two kinds: 1) beliefs about the world, with values such as true, false, possible, unlikely, etc.; and 2) judgments about the world, with values such as good, bad, neutral, wise, foolish, virtuous, etc. Statements like "I believe that he is smart" and "Stock prices will rise soon" are examples of beliefs whereas "I like the new policy on social security" and "Unfortunately this really was his year: despite a stagnant economy, he still won his re-election" are examples of judgment opinions. However, judgment opinions and beliefs are not necessarily mutually exclusive. For example, "I think it is an outrage" or "I believe that he is smart" carry both a belief and a judgment.

In the NIST pilot study, it was apparent that human annotators often disagreed on whether a belief statement was or was not an opinion. However, high annotator agreement was seen on judg-

[1] TREC novelty track 2003 and 2004

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 200–207,
New York, June 2006.

ment opinions. In this paper, we therefore focus our analysis on judgment opinions only. We hope that future work yields a more precise definition of belief opinions on which human annotators can agree.

We define a judgment opinion as consisting of three elements: a valence, a holder, and a topic. The *valence*, which applies specifically to judgment opinions and not beliefs, is the value of the judgment. In our framework, we consider the following valences: positive, negative, and neutral. The *holder* of an opinion is the person, organization or group whose opinion is expressed. Finally, the *topic* is the event or entity about which the opinion is held.

In previous work, Choi et al. (2005) identify opinion holders (sources) using Conditional Random Fields (CRF) and extraction patterns. They define the opinion holder identification problem as a sequence tagging task: given a sequence of words ($x_1 x_2 \cdots x_n$) in a sentence, they generate a sequence of labels ($y_1 y_2 \cdots y_n$) indicating whether the word is a holder or not. However, there are many cases where multiple opinions are expressed in a sentence each with its own holder. In those cases, finding opinion holders for each individual expression is necessary. In the corpus they used, 48.5% of the sentences which contain an opinion have more than one opinion expression with multiple opinion holders. This implies that multiple opinion expressions in a sentence occur significantly often. A major challenge of our work is therefore not only to focus on sentence with only one opinion, but also to identify opinion holders when there is more than one opinion expressed in a sentence. For example, consider the sentence "*In relation to Bush's axis of evil remarks, the German Foreign Minister also said, Allies are not satellites, and the French Foreign Minister caustically criticized that the United States' unilateral, simplistic worldview poses a new threat to the world*". Here, "*the German Foreign Minister*" should be the holder for the opinion "*Allies are not satellites*" and "*the French Foreign Minister*" should be the holder for "*caustically criticized*".

In this paper, we introduce a methodology for analyzing judgment opinions. We decompose the task into four parts: 1) recognizing the opinion; 2) identifying the valence; 3) identifying the holder; and 4) identifying the topic. For the purposes of this paper, we address the first three parts and leave the last for future work. Opinions can be extracted from various granularities such as a word, a sentence, a text, or even multiple texts. Each is important, but we focus our attention on word-level opinion detection (Section 2.1) and the detection of opinions in short emails (Section 3). We evaluate our methodology using intrinsic and extrinsic measures.

The remainder of the paper is organized as follows. In the next section, we describe our methodology addressing the three steps described above, and in Section 4 we present our experimental results. We conclude with a discussion of future work.

2 Analysis of Judgment Opinions

In this section, we first describe our methodology for detecting opinion bearing words and for identifying their valence, which is described in Section 2.1. Then, in Section 2.2, we describe our algorithm for identifying opinion holders. In Section 3, we show how to use our methodology for detecting opinions in short emails.

2.1 Detecting Opinion-Bearing Words and Identifying Valence

We introduce an algorithm to classify a word as being positive, negative, or neutral classes. This classifier can be used for any set of words of interest and the resulting words with their valence tags can help in developing new applications such as a public opinion monitoring system. We define an *opinion-bearing word* as a word that carries a positive or negative sentiment directly such as "good", "bad", "foolish", "virtuous", etc. In other words, this is the smallest unit of opinion that can thereafter be used as a clue for sentence-level or text-level opinion detection.

We treat word sentiment classification into Positive, Negative, and Neutral as a three-way classification problem instead of a two-way classification problem of Positive and Negative. By adding the third class, Neutral, we can prevent the classifier from assigning either positive or negative sentiment to weak opinion-bearing words. For example, the word "central" that Hatzivassiloglou and McKeown (1997) included as a positive adjective is not classified as positive in our system. Instead

we mark it as "neutral" since it is a weak clue for an opinion. If an unknown word has a strong relationship with the neutral class, we can therefore classify it as neutral even if it has some small connotation of Positive or Negative as well.

Approach: We built a word sentiment classifier using WordNet and three sets of positive, negative, and neutral words tagged by hand. Our insight is that synonyms of positive words tend to have positive sentiment. We expanded those manually selected seed words of each sentiment class by collecting synonyms from WordNet. However, we cannot simply assume that all the synonyms of positive words are positive since most words could have synonym relationships with all three sentiment classes. This requires us to calculate the closeness of a given word to each category and determine the most probable class. The following formula describes our model for determining the category of a word:

$$\arg\max_c P(c \mid w) \cong \arg\max_c P(c \mid syn_1, syn_2 ..... syn_n) \quad (1)$$

where c is a category (Positive, Negative, or Neutral) and w is a given word; syn_n is a WordNet synonym of the word w. We calculate this closeness as follows;

$$\arg\max_c P(c \mid w) = \arg\max_c P(c)P(w \mid c)$$
$$= \arg\max_c P(c)P(syn_1 syn_2 syn_3 ... syn_n \mid c)$$
$$= \arg\max_c P(c)\prod_{k=1}^{m} P(f_k \mid c)^{count(f_k, synset(w))} \quad (2)$$

where f_k is the k^{th} feature of class c which is also a member of the synonym set of the given word w. $count(f_k, synset(w))$ is the total number of occurrences of the word feature f_k in the synonym set of word w. In section 4.1, we describe our manually annotated dataset which we used for seed words and for our evaluation.

2.2 Identifying Opinion Holders

Despite successes in identifying opinion expressions and subjective words/phrases (See Section 1), there has been less achievement on the factors closely related to subjectivity and polarity, such as identifying the opinion holder. However, our research indicates that *without* this information, it is difficult, if not impossible, to define 'opinion' accurately enough to obtain reasonable inter-annotator agreement. Since these factors co-occur and mutually reinforce each other, the question "Who is the holder of this opinion?" is as important as "Is this an opinion?" or "What kind of opinion is expressed here?".

Sentence	Iraqi Vice President Taha Yassin Ramadan, responding to Bush's 'axis of evil' remark, said the U.S. government 'is the source of evil' in the world.
Expressive subjectivity	the U.S. government 'is the source of evil' in the world
Strength	Extreme
Source	Iraqi Vice President Taha Yassin Ramadan

Table 1: Annotation example

In this section, we describe the automated identification for opinion holders. We define an opinion holder as an entity (person, organization, country, or special group of people) who expresses explicitly or implicitly the opinion contained in the sentence.

Previous work that is related to opinion holder identification is (Bethard et al. 2004) who identify opinion propositions and holders. However, their opinion is restricted to propositional opinion and mostly to verbs. Another related work is (Choi et al. 2005) who use the MPQA corpus[2] to learn patterns of opinion sources using a graphical model and extraction pattern learning. However, they have a different task definition from ours. They define the task as identifying opinion sources (holders) given a sentence, whereas we define it as identifying opinion sources given an opinion expression in a sentence. We discussed their work in Section 1.

Data: As training data, we used the MPQA corpus (Wilson and Wiebe, 2003), which contains news articles manually annotated by 5 trained annotators. They annotated 10657 sentences from 535 documents, in four different aspects: *agent, expressive-subjectivity, on,* and *inside*. *Expressive-subjectivity* marks words and phrases that indirectly express a *private state* that is defined as a term for opinions, evaluations, emotions, and speculations. The *on* annotation is used to mark speech events and direct expressions of private states. As for the holder, we use the agent of the selected private states or speech events. While there are many possible ways to define what opinion means, intuitively, given an opinion, it is clear what the opinion holder means. Table 1 shows an example of the annotation. In this example, we consider the expression "the U.S. government 'is the source of evil' in the world" with an expres-

[2] http://www.cs.pitt.edu/~wiebe/pubs/ardasummer02/

sive-subjectivity tag as an opinion of the holder "Iraqi Vice President Taha Yassin Ramadan".

Approach: Since more than one opinion may be expressed in a sentence, we have to find an opinion holder for each opinion expression. For example, in a sentence "A thinks B's criticism of T is wrong", B is the holder of "the criticism of T", whereas A is the person who has an opinion that B's criticism is wrong. Therefore, we define our task as finding an opinion holder, given an opinion expression. Our earlier work (ref suppressed) focused on identifying opinion expressions within text. We employ that system in tandem with the one described here.

To learn opinion holders automatically, we use a Maximum Entropy model. Maximum Entropy models implement the intuition that the best model is the one that is consistent with the set of constraints imposed by the evidence but otherwise is as uniform as possible (Berger et al. 1996). There are two ways to model the problem with ME: classification and ranking. Classification allocates each holder candidate to one of a set of predefined classes while ranking selects a single candidate as answer. This means that classification modeling[3] can select many candidates as answers as long as they are marked as true, and does not select any candidate if every one is marked as false. In contrast, ranking always selects the most probable candidate as an answer, which suits our task better. Our earlier experiments showed poor performance with classification modeling, an experience also reported for Question Answering (Ravichandran et al. 2003).

We modeled the problem to choose the most probable candidate that maximizes a given conditional probability distribution, given a set of holder candidates $\{h_1 h_2 \ldots h_N\}$ and opinion expression *e*. The conditional probability $P(h \mid \{h_1 h_2 \ldots h_N\}, e)$ can be calculated based on *K* feature functions $f_k(h, \{h_1 h_2 \ldots h_N\}, e)$. We write a decision rule for the ranking as follows:

$$h = \underset{h}{\operatorname{argmax}} \quad [P(h \mid \{h_1 h_2 \ldots h_N\}, e)]$$

$$= \underset{h}{\operatorname{argmax}} \quad [\sum_{k=1}^{K} \lambda_k f_k(h, \{h_1 h_2 \ldots h_N\}, e)]$$

Each λ_k is a model parameter indicating the weight of its feature function.

[3] In our task, there are two classes: holder and non-holder.

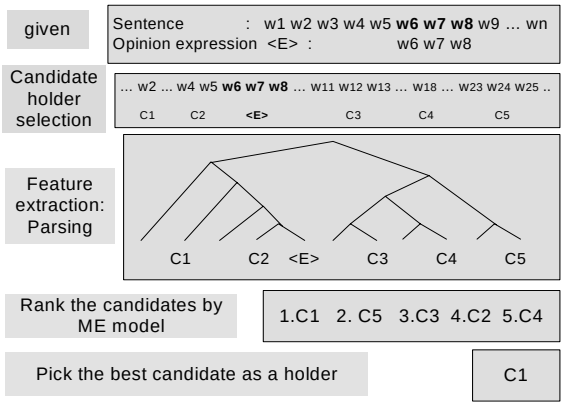


Figure 1: Overall system architecture

Figure 1 illustrates our holder identification system. First, the system generates all possible holder candidates, given a sentence and an opinion expression <E>. After parsing the sentence, it extracts features such as the syntactic path information between each candidate <H> and the expression <E> and a distance between <H> and <E>. Then it ranks holder candidates according to the score obtained by the ME ranking model. Finally the system picks the candidate with the highest score. Below, we describe in turn how to select holder candidates and how to select features for the training model.

Holder Candidate Selection: Intuitively, one would expect most opinion holders to be named entities (PERSON or ORGANIZATION)[4]. However, other common noun phrases can often be opinion holders, such as "the leader", "three nations", and "the Arab and Islamic world". Sometimes, pronouns like *he, she,* and *they* that refer to a PERSON, or *it* that refers to an ORGANIZATION or country, can be an opinion holder. In our study, we consider all noun phrases, including common noun phrases, named entities, and pronouns, as holder candidates.

Feature Selection: Our hypothesis is that there exists a structural relation between a holder <H> and an expression <E> that can help to identify opinion holders. This relation may be represented by lexical-level patterns between <H> and <E>, but anchoring on surface words might run into the data sparseness problem. For example, if we see the lexical pattern "<H> recently criticized <E>" in the training data, it is impossible to match the expression "<H> yesterday condemned <E>". These, however, have the same syntactic features in our

[4] We use BBN's named entity tagger IdentiFinder to collect named entities.

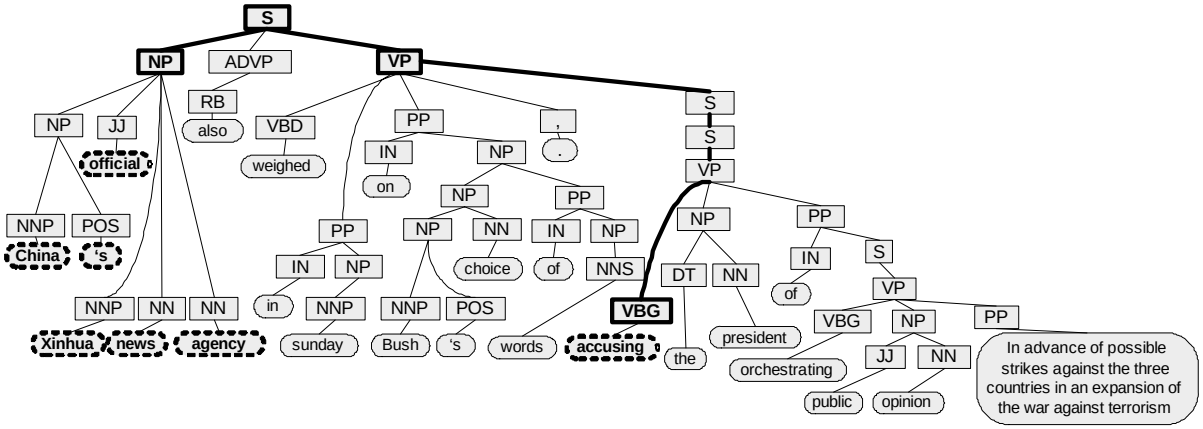


Figure 2: A parsing example

model. We therefore selected structural features from a deep parse, using the Charniak parser.

After parsing the sentence, we search for the *lowest common parent node* of the words in <H> and <E> respectively (<H> and <E> are mostly expressed with multiple words). A lowest common parent node is a non-terminal node in a parse tree that covers all the words in <H> and <E>. Figure 2 shows a parsed example of a sentence with the holder "China's official Xinhua news agency" and the opinion expression "accusing". In this example, the lowest common parent of words in <H> is the bold NP and the lowest common parent of <E> is the bold VBG. We name these nodes *Hhead* and *Ehead* respectively. After finding these nodes, we label them by subscript (e.g., NP_H and VBG_E) to indicate they cover <H> and <E>. In order to see how Hhead and Ehead are related to each other in the parse tree, we define another node, *HEhead,* which covers both Hhead and Ehead. In the example, HEhead is S at the top of the parse tree since it covers both NP_H and VBG_E. We also label S by subscript as S_{HE}.

To express tree structure for ME training, we extract path information between <H> and <E>. In the example, the complete path from Hhead to Ehead is "<H> NP S VP S S VP VBG <E>". However, representing each complete path as a single feature produces so many different paths with low frequencies that the ME system would learn poorly. Therefore, we split the path into three parts: *HEpath, Hpath an Epath*. HEpath is defined as a path from HEhead to its left and right child nodes that are also parents of Hhead and Ehead. Hpath is a path from Hhead and one of its ancestor nodes that is a child of HEhead. Similarly, Epath is defined as a path from Ehead to one of its ancestors that is also a child of HEhead. With this splitting, the system can work when any of HEpath, Hpath or Epath appeared in the training data, even if the entire path from <H> to <E> is unseen. Table 2 summarizes these concepts with two holder candidate examples in the parse tree of Figure 2.

We also include two non-structural features. The first is the type of the candidate, with values NP, PERSON, ORGANIZATION, and LOCATION. The second feature is the distance between <H> and <E>, counted in parse tree words. This is motivated by the intuition that holder candidates tend to lie closer to their opinion expression. All features are listed in Table 3. We describe the performance of the system in Section 4.

	Candidate 1	Candidate 2
	China's official Xinuhua news agency	Bush
Hhead	NP_H	NNP_H
Ehead	VBG_E	VBG_E
HEhead	S_{HE}	VP_{HE}
Hpath	NP_H	$NNP_H\ NP_H\ NP_H$ $NP_H\ PP_H$
Epath	$VBG_E\ VP_E\ S_E\ S_E\ VP_E$	$VBG_E\ VP_E\ S_E\ S_E$
HEpath	$S_{HE}\ NP_H\ VP_E$	$VP_{HE}\ PP_H\ S_E$

Table 2: Heads and paths for the Figure 2 example

Features	Description
F1	Type of <H>
F2	HEpath
F3	Hpath
F4	Epath
F5	Distance between <H> and <E>

Table 3: Features for ME training

Model 1
· Translate a German email to English · Apply English opinion-bearing words
Model 2
· Translate English opinion-bearing words to German · Analyze a German email using the German opinion-bearing words.

Table 4: Two models of German Email opinion analysis system

3 Applying our Methodology to German Emails

In this section, we describe a German email analysis system into which we included the opinion-bearing words from Section 2.1 to detect opinions expressed in emails. This system is part of a collaboration with the EU-funded project QUALEG (Quality of Service and Legitimacy in eGovernment) which aims at enabling local governments to manage their policies in a transparent and trustable way[5]. For this purpose, local governments should be able to measure the performance of the services they offer, by assessing the satisfaction of its citizens. This need makes a system that can monitor and analyze citizens' emails essential. The goal of our system is to classify emails as neutral or as bearing a positive or negative opinion.

To generate opinion bearing words, we ran the word sentiment classifier from Section 2.1 on 8011 verbs to classify them into 807 positive, 785 negative, and 6149 neutral. For 19748 adjectives, the system classified them into 3254 positive, 303 negative, and 16191 neutral. Since our opinion-bearing words are in English and our target system is in German, we also applied a statistical word alignment technique, GIZA++ [6] (Och and Ney 2000). Running it on version two of the European Parliament corpus, we obtained statistics for 678,340 German-English word pairs and 577,362 English-German word pairs. Obtaining these two lists of translation pairs allows us to convert English words to German, and German to English, without a full document translation system. To utilize our English opinion-bearing words in a German opinion analysis system, we developed two models, outlined in Table 4, each of which is triggered at different points in the system.

In both models, however, we still need to decide how to apply opinion-bearing words as clues to determine the sentiment of a whole email. Our previous work on sentence level sentiment classification (ref suppressed) shows that the presence of any negative words is a reasonable indication of a negative sentence. Since our emails are mostly short (the average number of words in each email is 19.2) and we avoided collecting weak negative opinion clue words, we hypothesize that our previous sentence sentiment classification study works on the email sentiment analysis. This implies that an email is negative if it contains more than certain number of strong negative words. We tune this parameter using our training data. Conversely, if an email contains mostly positive opinion-bearing words, we classify it as a positive email. We assign neutral if an email does not contain any strong opinion-bearing words.

Manually annotated email data was provided by our joint research site. This data contains 71 emails from citizens regarding a German festival. 26 of them contained negative complaints, for example, the lack of parking space, and 24 of them were positive with complimentary comments to the organization. The rest of them were marked as "questions" such as how to buy festival tickets, "only text" of simple comments, "fuzzy", and "difficult". So, we carried system experiments on positive and negative emails with precision and recall. We report system results in Section 4.

4 Experiment Results

In this section, we evaluate the three systems described in Sections 2 and 3: detecting opinion-bearing words and identifying valence, identifying opinion holders, and the German email opinion analysis system.

4.1 Detecting Opinion-bearing Words

We described a word classification system to detect opinion-bearing words in Section 2.1. To examine its effectiveness, we annotated 2011 verbs and 1860 adjectives, which served as a gold standard[7]. These words were randomly selected from a

[5] http://www.qualeg.eupm.net/my_spip/index.php
[6] http://www.fjoch.com/GIZA++.html

[7] Although nouns and adverbs may also be opinion-bearing, we focus only on verbs and adjectives for this study.

	Positive	Negative	Neutral	Total
Verb	69	151	1791	2011
Adjective	199	304	1357	1860

Table 5: Word distribution in our gold standard

		Precision	Recall	F-score
P	V	20.5% ± 3.5%	82.4% ± 7.5%	32.3% ± 4.6%
	A	32.4% ± 3.8%	75.5% ± 6.1%	45.1% ± 4.4%
X	V	97.2% ± 0.6%	77.6% ± 1.4%	86.3% ± 0.7%
	A	89.5% ± 1.7%	67.1% ± 2.7%	76.6% ± 2.1%
N	V	37.8% ± 4.9%	76.2% ± 8.0%	50.1% ± 5.6%
	A	60.0% ± 4.1%	78.5% ± 4.9%	67.8% ± 3.8%

Table 6: Precision, recall, and F-score on word valence categorization for Positive (P), Negative (N) and Neutral (X) verbs (V) and adjectives (A) (with 95% confidence intervals)

collection of 8011 English verbs and 19748 English adjectives. We use training data as seed words for the WordNet expansion part of our algorithm (described in Section 2.1). Table 5 shows the distribution of each semantic class. In both verb and adjective annotation, neutral class has much more words than the positive or negative classes.

We measured the precision, recall, and F-score of our system using 10-fold cross validation. Table 6 shows the results with 95% confidence bounds. Overall (combining positive, neutral and negative), our system achieved 77.7% ± 1.2% accuracy on verbs and 69.1% ± 2.1% accuracy on adjectives. The system has very high precision in the neutral category for both verbs (97.2%) and adjectives (89.5%), which we interpret to mean that our system is really good at filtering non-opinion bearing words. Recall is high in all cases but precision varies; very high for neutral and relatively high for negative but low for positive.

4.2 Opinion Holder Identification

We conducted experiments on 2822 <sentence; opinion expression; holder> triples and divided the data set into 10 <training; test> sets for cross validation. For evaluation, we consider to match either fully or partially with the holder marked in the test data. The holder matches *fully* if it is a single entity (e.g., "Bush"). The holder matches *partially* when it is part of the multiple entities that make up the marked holder. For example, given a marked holder "Michel Sidibe, Director of the Country and Regional Support Department of UNAIDS", we consider both "Michel Sidibe" and "Director of the Country and Regional Support Department of UNAIDS" as acceptable answers.

	Baseline	**F5**	**F15**	**F234**	**F12345**
Top1	23.2%	21.8%	41.6%	50.8%	52.7%
Top2		39.7%	61.9%	66.3%	67.9%
Top3		52.2%	72.5%	77.1%	77.8%

Table 7: Opinion holder identification results (excluding pronouns from candidates)

	Baseline	**F5**	**F15**	**F234**	**F12345**
Top1	21.3%	18.9%	41.8%	47.9%	50.6%
Top2		37.9%	61.6%	64.8%	66.7%
Top3		51.2%	72.3%	75.3%	76.0%

Table 8: Opinion holder identification results (All noun phrases as candidates)

Our experiments consist of two parts based on the candidate selection method. Besides the selection method we described in Section 2.2, we also conducted a separate experiment by excluding pronouns from the candidate list. With the second method, the system always produces a non-pronoun holder as an answer. This selection method is useful in some Information Extraction application that only cares non-pronoun holders.

We report accuracy (the percentage of correct answers the system found in the test set) to evaluate our system. We also report how many correct answers were found within the top2 and top3 system answers. Tables 7 and 8 show the system accuracy with and without considering pronouns as alias candidates, respectively. Table 8 mostly shows lower accuracies than Table 7 because test data often has only a non-pronoun entity as a holder and the system picks a pronoun as its answer. Even if the pronoun refers the same entity marked in the test data, the evaluation system counts it as wrong because it does not match the hand annotated holder.

To evaluate the effectiveness of our system, we set the baseline as a system choosing the closest candidate to the expression as a holder without the Maximum Entropy decision. The baseline system had an accuracy of only 21.3% for candidate selection over all noun phrases and 23.2% for candidate selection excluding pronouns.

The results show that detecting opinion holders is a hard problem, but adopting syntactic features (F2, F3, and F4) helps to improve the system. A promising avenue of future work is to investigate the use of semantic features to eliminate noun

phrases such as “cheap energy subsidies” or “possible strikes” from the candidate set before we run our ME model, since they are less likely to be an opinion holder than noun phrases like “three nations” or “Palestine people.”

4.3 German Emails

For our experiment, we performed 7-fold cross validation on a set of 71 emails. Table 9 shows the average precision, recall, and F-score. Results show that our system identifies negative emails (complaints) better than praise. When we chose a system parameter for the focus, we intended to find negative emails rather than positive emails because officials who receive these emails need to act to solve problems when people complain but they have less need to react to compliments. By highlighting high recall of negative emails, we may misclassify a neutral email as negative but there is also less chance to neglect complaints.

Category		Model1	Model2
Positive (P)	Precision	0.72	0.55
	Recall	0.40	0.65
	F-score	0.51	0.60
Negative (N)	Precision	0.55	0.61
	Recall	0.80	0.42
	F-score	0.65	0.50

Table 9: German email opinion analysis system results

5 Conclusion and Future Work

In this paper, we presented a methodology for analyzing judgment opinions, which we define as opinions consisting of a valence, a holder, and a topic. We presented models for recognizing sentences containing judgment opinions, identifying the valence of the opinion, and identifying the holder of the opinion. Remaining is to also finally identify the topic of the opinion. Past tests with human annotators indicate that the accuracy of identifying valence, holder and topic is much increased when all three are being done simultaneously. We plan to investigate a joint model to verify this intuition.

Our past work indicated that, for newspaper texts, it is feasible for annotators to identify judgment opinion sentences and for them to identify their holders and judgment valences. It is encouraging to see that we achieved good results on a new genre – emails sent from citizens to a city council – and in a new language, German.

This paper presents a computational framework for analyzing judgment opinions. Even though these are the most common opinions, it is a pity that the research community remains unable to define *belief* opinions (i.e., those opinions that have values such as true, false, possible, unlikely, etc.) with high enough inter-annotator agreement. Only once we properly define belief opinion will we be capable of building a complete opinion analysis system.

References

Berger, A, S. Della Pietra, and V. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language. *Computational Linguistics 22(1).*

Bethard, S., H. Yu, A. Thornton, V. Hatzivassiloglou, and D. Jurafsky. 2004. Automatic Extraction of Opinion Propositions and their Holders. *AAAI Spring Symposium on Exploring Attitude and Affect in Text.*

Charniak, E. 2000. A Maximum-Entropy-Inspired Parser. *Proc. of NAACL-2000.*

Choi, Y., C. Cardie, E. Riloff, and S. Patwardhan. 2005. Identifying Sources of Opinions with Conditional Random Fields and Extraction Patterns. *Proc. of Human Language Technology Conference/Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP 2005).*

Esuli, A. and F. Sebastiani. 2005. Determining the semantic orientation of terms through gloss classification. *Proc. of CIKM-05, 14th ACM International Conference on Information and Knowledge Management.*

Hatzivassiloglou, V. and McKeown, K. (1997). Predicting the semantic orientation of adjectives. *Proc. 35th Annual Meeting of the Assoc. for Computational Linguistics (ACL-EACL 97.*

Hu, M. and Liu, B. 2004. Mining and summarizing customer reviews. *Proc. of KDD'04. pp.168 - 177*

Och, F.J. 2002. Yet Another MaxEnt Toolkit: YASMET *http://wasserstoff.informatik.rwth-aachen.de/Colleag ues/och/*

Och, F.J and Ney, H. 2000. Improved statistical alignment models. *Proc. of ACL-2000, pp. 440–447, Hong Kong, China.*

Pang, B, L. Lee, and S. Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. *Proc. of EMNLP 2002.*

Ravichandran, D., E. Hovy, and F.J. Och. 2003. Statistical QA - classifier vs re-ranker: What's the difference? *Proc. of the ACL Workshop on Multilingual Summarization and Question Answering*.

Riloff, E. and J. Wiebe. 2003. Learning Extraction Patterns for Subjective Expressions. *Proc. of EMNLP-03*.

Turney, P. 2002. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. *Proc. of the 40th Annual Meeting of the ACL, 417–424.*

Wiebe, J, R. Bruce, and T. O'Hara. 1999. Development and use of a gold standard data set for subjectivity classifications. *Proc. of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99), 246–253.*

Wilson, T. and J. Wiebe. 2003. Annotating Opinions in the World Press. *Proc. of ACL SIGDIAL-03.*

Yu, H. and V. Hatzivassiloglou. 2003. Towards Answering Opinion Questions: Separating Facts from Opinions and Identifying the Polarity of Opinion Sentences. *Proc. of EMNLP.*

Learning to Detect Conversation Focus of Threaded Discussions

Donghui Feng **Erin Shaw** **Jihie Kim** **Eduard Hovy**
Information Sciences Institute
University of Southern California
Marina del Rey, CA, 90292
{donghui, shaw, jihie, hovy}@isi.edu

Abstract

In this paper we present a novel feature-enriched approach that learns to detect the conversation focus of threaded discussions by combining NLP analysis and IR techniques. Using the graph-based algorithm HITS, we integrate different features such as lexical similarity, poster trustworthiness, and speech act analysis of human conversations with feature-oriented link generation functions. It is the first quantitative study to analyze human conversation focus in the context of online discussions that takes into account heterogeneous sources of evidence. Experimental results using a threaded discussion corpus from an undergraduate class show that it achieves significant performance improvements compared with the baseline system.

1 Introduction

Threaded discussion is popular in virtual cyber communities and has applications in areas such as customer support, community development, interactive reporting (blogging) and education. Discussion threads can be considered a special case of human conversation, and since we have huge repositories of such discussion, automatic and/or semi-automatic analysis would greatly improve the navigation and processing of the information.

A discussion thread consists of a set of messages arranged in chronological order. One of the main challenges in the Question Answering domain is how to extract the most informative or important message in the sequence for the purpose of answering the initial question, which we refer to as the *conversation focus* in this paper. For example, people may repeatedly discuss similar questions in a discussion forum and so it is highly desirable to detect previous conversation focuses in order to automatically answer queries (Feng et al., 2006).

Human conversation focus is a hard NLP (Natural Language Processing) problem in general because people may frequently switch topics in a real conversation. The threaded discussions make the problem manageable because people typically focus on a limited set of issues within a thread of a discussion. Current IR (Information Retrieval) techniques are based on keyword similarity measures and do not consider some features that are important for analyzing threaded discussions. As a result, a typical IR system may return a ranked list of messages based on keyword queries even if, within the context of a discussion, this may not be useful or correct.

Threaded discussion is a special case of human conversation, where people may express their ideas, elaborate arguments, and answer others' questions; many of these aspects are unexplored by traditional IR techniques. First, messages in threaded discussions are not a flat document set, which is a common assumption for most IR systems. Due to the flexibility and special characteristics involved in human conversations, messages within a thread are not necessarily of equal importance. The real relationships may differ from the analysis based on keyword similarity measures, e.g., if a 2nd message "corrects" a 1st one, the 2nd message is probably more important than the 1st. IR systems may give different results. Second, messages posted by different users may have different degrees of correctness and trustworthiness, which we refer to as *poster trustworthiness* in this paper. For instance, a domain expert is likely to be more reliable than a layman on the domain topic.

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 208–215,
New York, June 2006.

In this paper we present a novel feature-enriched approach that learns to detect conversation focus of threaded discussions by combining NLP analysis and IR techniques. Using the graph-based algorithm HITS (Hyperlink Induced Topic Search, Kleinberg, 1999), we conduct discussion analysis taking into account different features, such as lexical similarity, poster trustworthiness, and speech act relations in human conversations. We generate a weighted threaded discussion graph by applying feature-oriented link generation functions. All the features are quantified and integrated as part of the weight of graph edges. In this way, both quantitative features and qualitative features are combined to analyze human conversations, specifically in the format of online discussions.

To date, it is the first quantitative study to analyze human conversation that focuses on threaded discussions by taking into account heterogeneous evidence from different sources. The study described here addresses the problem of conversation focus, especially for extracting the best answer to a particular question, in the context of an online discussion board used by students in an undergraduate computer science course. Different features are studied and compared when applying our approach to discussion analysis. Experimental results show that performance improvements are significant compared with the baseline system.

The remainder of this paper is organized as follows: We discuss related work in Section 2. Section 3 presents thread representation and the weighted HITS algorithm. Section 4 details feature-oriented link generation functions. Comparative experimental results and analysis are given in Section 5. We discuss future work in Section 6.

2 Related Work

Human conversation refers to situations where two or more participants freely alternate in speaking (Levinson, 1983). What makes threaded discussions unique is that users participate asynchronously and in writing. We model human conversation as a set of messages in a threaded discussion using a graph-based algorithm.

Graph-based algorithms are widely applied in link analysis and for web searching in the IR community. Two of the most prominent algorithms are Page-Rank (Brin and Page, 1998) and the HITS algorithm (Kleinberg, 1999). Although they were initially proposed for analyzing web pages, they proved useful for investigating and ranking structured objects. Inspired by the idea of graph based algorithms to collectively rank and select the best candidate, research efforts in the natural language community have applied graph-based approaches on keyword selection (Mihalcea and Tarau, 2004), text summarization (Erkan and Radev, 2004; Mihalcea, 2004), word sense disambiguation (Mihalcea et al., 2004; Mihalcea, 2005), sentiment analysis (Pang and Lee, 2004), and sentence retrieval for question answering (Otterbacher et al., 2005). However, until now there has not been any published work on its application to human conversation analysis specifically in the format of threaded discussions. In this paper, we focus on using HITS to detect conversation focus of threaded discussions.

Rhetorical Structure Theory (Mann and Thomson, 1988) based discourse processing has attracted much attention with successful applications in sentence compression and summarization. Most of the current work on discourse processing focuses on sentence-level text organization (Soricut and Marcu, 2003) or the intermediate step (Sporleder and Lapata, 2005). Analyzing and utilizing discourse information at a higher level, e.g., at the paragraph level, still remains a challenge to the natural language community. In our work, we utilize the discourse information at a message level.

Zhou and Hovy (2005) proposed summarizing threaded discussions in a similar fashion to multi-document summarization; but then their work does not take into account the relative importance of different messages in a thread. Marom and Zukerman (2005) generated help-desk responses using clustering techniques, but their corpus is composed of only two-party, two-turn, conversation pairs, which precludes the need to determine relative importance as in a multi-ply conversation.

In our previous work (Feng et al., 2006), we implemented a discussion-bot to automatically answer student queries in a threaded discussion but extract potential answers (the most informative message) using a rule-based traverse algorithm that is not optimal for selecting a best answer; thus, the result may contain redundant or incorrect information. We argue that pragmatic knowledge like speech acts is important in conversation focus analysis. However, estimated speech act labeling between messages is not sufficient for detecting

human conversation focus without considering other features like author information. Carvalho and Cohen (2005) describe a dependency-network based collective classification method to classify email speech acts. Our work on conversation focus detection can be viewed as an immediate step following automatic speech act labeling on discussion threads using similar collective classification approaches.

We next discuss our approach to detect conversation focus using the graph-based algorithm HITS by taking into account heterogeneous features.

3 Conversation Focus Detection

In threaded discussions, people participate in a conversation by posting messages. Our goal is to be able to detect which message in a thread contains the most important information, i.e., the *focus* of the conversation. Unlike traditional IR systems, which return a ranked list of messages from a flat document set, our task must take into account characteristics of threaded discussions.

First, messages play certain roles and are related to each other *by a conversation context*. Second, messages written by different authors may *vary in value*. Finally, since postings occur in parallel, by various people, message threads are not necessarily coherent so the *lexical similarity* among the messages should be analyzed. To detect the focus of conversation, we integrate a pragmatics study of conversational speech acts, an analysis of message values based on poster trustworthiness and an analysis of lexical similarity. The subsystems that determine these three sources of evidence comprise the features of our feature-based system.

Because each discussion thread is naturally represented by a directed graph, where each message is represented by a node in the graph, we can apply a graph-based algorithm to integrate these sources and detect the focus of conversation.

3.1 Thread Representation

A discussion thread consists of a set of messages posted in chronological order. Suppose that each message is represented by m_i, $i = 1,2,\ldots, n$. Then the entire thread is a directed graph that can be represented by $G = (V, E)$, where V is the set of nodes (messages), $V = \{m_i, i=1,\ldots,n\}$, and E is the set of directed edges. In our approach, the set V is automatically constructed as each message joins in the discussion. E is a subset of VxV. We will discuss the feature-oriented link generation functions that construct the set E in Section 4.

We make use of speech act relations in generating the links. Once a speech act relation is identified between two messages, links will be generated using generation functions described in next section. When m_i is a message node in the thread graph, $F(m_i) \subset V$ represents the set of nodes that node m_i points to (i.e., children of m_i), and $B(m_i) \subset V$ represents the set of nodes that point to m_i (i.e., parents of m_i).

3.2 Graph-Based Ranking Algorithm: HITS

Graph-based algorithms can rank a set of objects in a collective way and the affect between each pair can be propagated into the whole graph iteratively. Here, we use a weighted HITS (Kleinberg, 1999) algorithm to conduct message ranking.

Kleinberg (1999) initially proposed the graph-based algorithm HITS for ranking a set of web pages. Here, we adjust the algorithm for the task of ranking a set of messages in a threaded discussion. In this algorithm, each message in the graph can be represented by two identity scores, *hub score* and *authority score*. The hub score represents the quality of the message as a pointer to valuable or useful messages (or resources, in general). The authority score measures the quality of the message as a resource itself. The weighted iterative updating computations are shown in Equations 1 and 2.

$$hub^{r+1}(m_i) = \sum_{m_j \in F(m_i)} w_{ij} * authority^{r}(m_j) \qquad (1)$$

$$authority^{r+1}(m_i) = \sum_{m_j \in B(m_i)} w_{ji} * hub^{r}(m_j) \qquad (2)$$

where r and $r+1$ are the numbers of iterations.

The number of iterations required for HITS to converge depends on the initialization value for each message node and the complexity of the graph. Graph links can be induced with extra knowledge (e.g. Kurland and Lee, 2005). To help integrate our heterogeneous sources of evidence with our graph-based HITS algorithm, we introduce link generation functions for each of the three features, (g_i, $i=1, 2, 3$), to add links between messages.

4 Feature-Oriented Link Generation

Conversation structures have received a lot of attention in the linguistic research community (Levinson, 1983). In order to integrate conversational features into our computational model, we must convert a qualitative analysis into quantitative scores. For conversation analysis, we adopted the theory of Speech Acts proposed by (Austin, 1962; Searle, 1969) and defined a set of speech acts (SAs) that relate every pair of messages in the corpus. Though a pair of messages may only be labeled with one speech act, a message can have multiple SAs with other messages.

We group speech acts by function into three categories, as shown in Figure 1. Messages may involve a request (REQ), provide information (INF), or fall into the category of interpersonal (INTP) relationship. Categories can be further divided into several single speech acts.

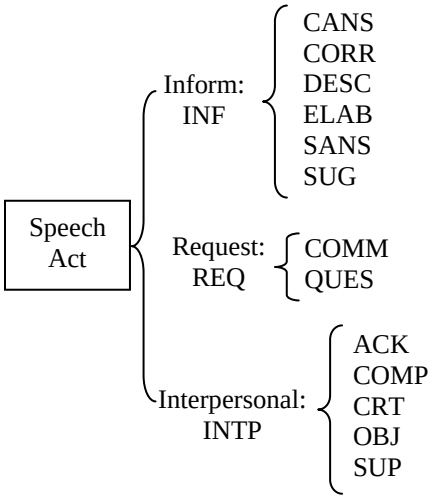


Figure 1. Categories of Message Speech Act.

The SA set for our corpus is given in Table 1. A speech act may a represent a positive, negative or neutral response to a previous message depending on its attitude and recommendation. We classify each speech act as a direction as POSITIVE (+), NEGATIVE (−) or NEUTRAL, referred to as *SA Direction*, as shown in the right column of Table 1.

The features we wish to include in our approach are lexical similarity between messages, poster trustworthiness, and speech act labels between message pairs in our discussion corpus.

The feature-oriented link generation is conducted in two steps. First, our approach examines in turn all the speech act relations in each thread and generates two types of links based on lexical similarity and SA strength scores. Second, the system iterates over all the message nodes and assigns each node a self-pointing link associated with its poster trustworthiness score. The three features are integrated into the thread graph accordingly by the feature-oriented link generation functions. Multiple links with the same start and end points are combined into one.

Speech Act	Name	Description	Dir.
ACK	Acknowl-edge	Confirm or acknowledge	+
CANS	Complex Answer	Give answer requiring a full description of procedures, reasons, etc.	
COMM	Command	Command or announce	
COMP	Compli-ment	Praise an argument or suggestion	+
CORR	Correct	Correct a wrong answer or solution	–
CRT	Criticize	Criticize an argument	–
DESC	Describe	Describe a fact or situation	
ELAB	Elaborate	Elaborate on a previous argument or question	
OBJ	Object	Object to an argument or suggestion	–
QUES	Question	Ask question about a specific problem	
SANS	Simple Answer	Answer with a short phrase or few words (e.g. factoid, yes/no)	
SUG	Suggest	Give advice or suggest a solution	
SUP	Support	Support an argument or suggestion	+

Table 1. Types of message speech acts in corpus.

4.1 Lexical Similarity

Discussions are constructed as people express ideas, opinions, and thoughts, so that the text itself contains information about what is being discussed. Lexical similarity is an important measure for distinguishing relationships between message pairs. In our approach, we do not compute the lexical similarity of any arbitrary pair of messages, instead, we consider only message pairs that are present in the speech act set. The cosine similarity between each message pair is computed using the TF*IDF technique (Salton, 1989).

Messages with similar words are more likely to be semantically-related. This information is represented by term frequency (TF). However, those

with more general terms may be unintentionally biased when only TF is considered so Inverse Document Frequency (IDF) is introduced to mitigate the bias. The lexical similarity score can be calculated using their cosine similarity.

$$W^l = \cos\_sim(m_i, m_j) \quad (3)$$

For a given a speech act, $SA_{ij}(m_i \rightarrow m_j)$, connecting message m_i and m_j, the link generation function g_1 is defined as follows:

$$g_1(SA_{ij}) = arc_{ij}(W^l) \quad (4)$$

The new generated link is added to the thread graph connecting message node m_i and m_j with a weight of W^l.

4.2 Poster Trustworthiness

Messages posted by different people may have different degrees of trustworthiness. For example, students who contributed to our corpus did not seem to provide messages of equal value. To determine the trustworthiness of a person, we studied the responses to their messages throughout the entire corpus. We used the percentage of POSITIVE responses to a person's messages to measure that person's trustworthiness. In our case, POSITIVE responses, which are defined above, included SUP, COMP, and ACK. In addition, if a person's message closed a discussion, we rated it POSITIVE.

Suppose the poster is represented by $person_k$, the poster score, W^p, is a weight calculated by

$$W^p(person_k) = \frac{count(positive\_feedback(person_k))}{count(feedback(person_k))} \quad (5)$$

For a given single speech act, $SA_{ij}(m_i \rightarrow m_j)$, the poster score indicates the importance of message m_i by itself and the generation function is given by

$$g_2(SA_{ij}) = arc_{ii}(W^p) \quad (6)$$

The generated link is self-pointing, and contains the strength of the poster information.

4.3 Speech Act Analysis

We compute the strength of each speech act in a generative way, based on the author and trustworthiness of the author. The strength of a speech act is a weighted average over all authors.

$$W^s(SA) = sign(dir) \sum_{person_k} \frac{count(SA_{person_k})}{count(SA)} W^P(person_k) \quad (7)$$

where the sign function of *direction* is defined with Equation 8.

$$sign(dir) = \begin{cases} -1 & \text{if dir is NEGATIVE} \\ 1 & \text{Otherwise} \end{cases} \quad (8)$$

All SA scores are computed using Equation 7 and projected to [0, 1]. For a given speech act, $SA_{ij}(m_i \rightarrow m_j)$, the generation function will generate a weighted link in the thread graph as expressed in Equation 9.

$$g_3(SA_{ij}) = \begin{cases} arc_{ii}(W^s) & \text{if } SA_{ij} \text{ is NEUTRAL} \\ arc_{ij}(W^s) & \text{Otherwise} \end{cases} \quad (9)$$

The SA scores represent the strength of the relationship between the messages. Depending on the direction of the SA, the generated link will either go from message m_i to m_j or from message m_i to m_i (i.e., to itself). If the *SA* is NEUTRAL, the link will point to itself and the score is a recommendation to itself. Otherwise, the link connects two different messages and represents the recommendation degree of the parent to the child message.

5 Experiments

5.1 Experimental Setup

We tested our conversation-focus detection approach using a corpus of threaded discussions from three semesters of a USC undergraduate course in computer science. The corpus includes a total of 640 threads consisting of 2214 messages, where a thread is defined as an exchange containing at least two messages.

Length of thread	Number of threads
3	139
4	74
5	47
6	30
7	13
8	11

Table 2. Thread length distribution.

From the complete corpus, we selected only threads with lengths of greater than two and less than nine (messages). Discussion threads with lengths of only two would bias the random guess of our baseline system, while discussion threads with lengths greater than eight make up only 3.7% of the total number of threads (640), and are the least coherent of the threads due to topic-switching and off-topic remarks. Thus, our evaluation corpus included 314 threads, consisting of 1307 messages, with an average thread length of 4.16 messages per

thread. Table 2 gives the distribution of the lengths of the threads.

The input of our system requires the identification of speech act relations between messages. Collective classification approaches, similar to the dependency-network based approach that Carvalho and Cohen (2005) used to classify email speech acts, might also be applied to discussion threads. However, as the paper is about investigating how an SA analysis, along with other features, can benefit conversation focus detection, so as to avoid error propagation from speech act labeling to subsequent processing, we used manually-annotated SA relationships for our analysis.

Code	Frequency	Percentage (%)
ACK	53	3.96
CANS	224	16.73
COMM	8	0.6
COMP	7	0.52
CORR	20	1.49
CRT	23	1.72
DESC	71	5.3
ELAB	105	7.84
OBJ	21	1.57
QUES	450	33.61
SANS	23	1.72
SUG	264	19.72
SUP	70	5.23

Table 3. Frequency of speech acts.

The corpus contains 1339 speech acts. Table 3 gives the frequencies and percentages of speech acts found in the data set. Each SA generates feature-oriented weighted links in the threaded graph accordingly as discussed previously.

Number of best answers	Number of threads
1	250
2	56
3	5
4	3

Table 4. Gold standard length distribution.

We then read each thread and choose the message that contained the best answer to the initial query as the gold standard. If there are multiple best-answer messages, all of them will be ranked as best, i.e., chosen for the top position. For example, different authors may have provided suggestions that were each correct for a specified situation. Table 4 gives the statistics of the numbers of correct messages of our gold standard.

We experimented with further segmenting the messages so as to narrow down the best-answer text, under the assumption that long messages probably include some less-than-useful information. We applied TextTiling (Hearst, 1994) to segment the messages, which is the technique used by Zhou and Hovy (2005) to summarize discussions. For our corpus, though, the ratio of segments to messages was only 1.03, which indicates that our messages are relatively short and coherent, and that segmenting them would not provide additional benefits.

5.2 Baseline System

To compare the effectiveness of our approach with different features, we designed a baseline system that uses a random guess approach. Given a discussion thread, the baseline system randomly selects the most important message. The result was evaluated against the gold standard. The performance comparisons of the baseline system and other feature-induced approaches are presented next.

5.3 Result Analysis and Discussion

We conducted extensive experiments to investigate the performance of our approach with different combinations of features. As we discussed in Section 4.2, each poster acquires a trustworthiness score based on their behavior via an analysis of the whole corpus. Table 5 is a sample list of some posters with their poster id, the total number of responses (to their messages), the total number of positive responses, and their poster scores W^p.

Poster ID	Total Response	Positive Response	W^p
193	1	1	1
93	20	18	0.9
38	15	12	0.8
80	8	6	0.75
47	253	182	0.719
22	3	2	0.667
44	9	6	0.667
91	6	4	0.667
147	12	8	0.667
32	10	6	0.6
190	9	5	0.556
97	20	11	0.55
12	2	1	0.5

Table 5. Sample poster scores.

Based on the poster scores, we computed the strength score of each SA with Equation 7 and projected them to [0, 1]. Table 6 shows the strength scores for all of the SAs. Each SA has a different strength score and those in the NEGATIVE category have smaller ones (weaker recommendation).

SA	$W^s(SA)$	SA	$W^s(SA)$
CANS	0.8134	COMM	0.6534
DESC	0.7166	ELAB	0.7202
SANS	0.8281	SUG	0.8032
QUES	0.6230		
ACK	0.6844	COMP	0.8081
SUP	0.8057		
CORR	0.2543	CRT	0.1339
OBJ	0.2405		

Table 6. SA strength scores.

We tested the graph-based HITS algorithm with different feature combinations and set the error rate to be 0.0001 to get the algorithm to converge. In our experiments, we computed the precision score and the MRR (Mean Reciprocal Rank) score (Voorhees, 2001) of the most informative message chosen (the first, if there was more than one). Table 7 shows the performance scores for the system with different feature combinations. The performance of the baseline system is shown at the top.

The HITS algorithm assigns both a hub score and an authority score to each message node, resulting in two sets of results. Scores in the HITS_AUTHORITY rows of Table 7 represent the results using authority scores, while HITS_HUB rows represent the results using hub scores.

Due to the limitation of thread length, the lower bound of the MRR score is 0.263. As shown in the table, a random guess baseline system can get a precision of 27.71% and a MRR score of 0.539.

When we consider only lexical similarity, the result is not so good, which supports the notion that in human conversation context is often more important than text at a surface level. When we consider poster and lexical score together, the performance improves. As expected, the best performances use speech act analysis. More features do not always improve the performance, for example, the lexical feature will sometimes decrease performance. Our best performance produced a precision score of 70.38% and an MRR score of 0.825, which is a significant improvement over the baseline's precision score of 27.71% and its MRR score of 0.539.

Algorithm & Features		Correct (out of 314)	Precision (%)	MRR
Baseline		87	27.71	0.539
HITS_AUTHORITY	Lexical	65	20.70	0.524
	Poster	90	28.66	0.569
	SA	215	68.47	0.819
	Lexical + Poster	91	28.98	0.565
	Lexical + SA	194	61.78	0.765
	Poster + SA	**221**	**70.38**	**0.825**
	Lexical + Poster + SA	212	67.52	0.793
HITS_HUB	Lexical	153	48.73	0.682
	Poster	79	25.16	0.527
	SA	195	62.10	0.771
	Lexical + Poster	158	50.32	0.693
	Lexical + SA	177	56.37	0.724
	Poster + SA	**207**	**65.92**	**0.793**
	Lexical + Poster + SA	196	62.42	0.762

Table 7. System Performance Comparison.

Another widely-used graph algorithm in IR is PageRank (Brin and Page, 1998). It is used to investigate the connections between hyperlinks in web page retrieval. PageRank uses a "random walk" model of a web surfer's behavior. The surfer begins from a random node m_i and at each step either follows a hyperlink with the probability of d, or jumps to a random node with the probability of $(1\text{-}d)$. A weighted PageRank algorithm is used to model weighted relationships of a set of objects. The iterative updating expression is

$$PR^{r+1}(m_i) = (1-d) + d * \sum_{m_j \in B(m_i)} \frac{w_{ji}}{\sum_{m_k \in F(m_j)} w_{jk}} PR^r(m_j) \quad (10)$$

where r and $r+1$ are the numbers of iterations.

We also tested this algorithm in our situation, but the best performance had a precision score of only 47.45% and an MRR score of 0.669. It may be that PageRank's definition and modeling approach does not fit our situation as well as the HITS approach. In HITS, the authority and hub-

based approach is better suited to human conversation analysis than PageRank, which only considers the contributions from backward links of each node in the graph.

6 Conclusions and Future Work

We have presented a novel feature-enriched approach for detecting conversation focus of threaded discussions for the purpose of answering student queries. Using feature-oriented link generation and a graph-based algorithm, we derived a unified framework that integrates heterogeneous sources of evidence. We explored the use of speech act analysis, lexical similarity and poster trustworthiness to analyze discussions.

From the perspective of question answering, this is the first attempt to automatically answer complex and contextual discussion queries beyond factoid or definition questions. To fully automate discussion analysis, we must integrate automatic SA labeling together with our conversation focus detection approach. An automatic system will help users navigate threaded archives and researchers analyze human discussion.

Supervised learning is another approach to detecting conversation focus that might be explored. The tradeoff and balance between system performance and human cost for different learning algorithms is of great interest. We are also exploring the application of graph-based algorithms to other structured-objects ranking problems in NLP so as to improve system performance while relieving human costs.

Acknowledgements

The work was supported in part by DARPA grant DOI-NBC Contract No. NBCHC050051, *Learning by Reading*, and in part by a grant from the Lord Corporation Foundation to the USC Distance Education Network. The authors want to thank Deepak Ravichandran, Feng Pan, and Rahul Bhagat for their helpful suggestions with the manuscript. We would also like to thank the HLT-NAACL reviewers for their valuable comments.

References

Austin, J. 1962. *How to do things with words*. Cambridge, Massachusetts: Harvard Univ. Press.

Brin, S. and Page, L. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107--117.

Carvalho, V.R. and Cohen, W.W. 2005. On the collective classification of email speech acts. *In Proceedings of SIGIR-2005*, pp. 345-352.

Erkan, G. and Radev, D. 2004. Lexrank: graph-based centrality as salience in text summarization. *Journal of Artificial Intelligence Research (JAIR)*.

Feng, D., Shaw, E., Kim, J., and Hovy, E.H. 2006. An intelligent discussion-bot for answering student queries in threaded discussions. In *Proceedings of Intelligent User Interface (IUI-2006)*, pp. 171-177.

Hearst, M.A. 1994. Multi-paragraph segmentation of expository text. In *Proceedings of ACL-1994*.

Kleinberg, J. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5).

Kurland, O. and Lee L. 2005. PageRank without hyperlinks: Structural re-ranking using links induced by language models. In *Proceedings of SIGIR-2005*.

Levinson, S. 1983. *Pragmatics*. Cambridge Univ. Press.

Mann, W.C. and Thompson, S.A. 1988. Rhetorical structure theory: towards a functional theory of text organization. *Text*, 8 (3), pp. 243-281.

Marom, Y. and Zukerman, I. 2005. Corpus-based generation of easy help-desk responses. *Technical Report, Monash University*. Available at: http://www.csse.monash.edu.au/publications/2005/tr-2005-166-full.pdf.

Mihalcea, R. 2004. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Companion Volume to ACL-2004*.

Mihalcea, R. 2005. unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *HLT/EMNLP 2005*.

Mihalcea, R. and Tarau, P. 2004. TextRank: bringing order into texts. In *Proceedings of EMNLP 2004*.

Mihalcea, R., Tarau, P. and Figa, E. 2004. PageRank on semantic networks, with application to word sense disambiguation. In *Proceedings of COLING 2004*.

Otterbacher, J., Erkan, G., and Radev, D. 2005. Using random walks for question-focused sentence retrieval. In *Proceedings of HLT/EMNLP 2005*.

Pang, B. and Lee, L. 2004. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL-2004*.

Salton, G. 1989. *Automatic Text Processing, The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, Reading, MA, 1989.

Searle, J. 1969. *Speech Acts*. Cambridge: Cambridge Univ. Press.

Soricut, R. and Marcu, D. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of HLT/NAACL-2003*.

Sporleder, C. and Lapata, M. 2005. Discourse chunking and its application to sentence compression. In *Proceedings of HLT/EMNLP 2005*.

Voorhees, E.M. 2001. Overview of the TREC 2001 question answering track. In *TREC 2001*.

Zhou, L. and Hovy, E.H. 2005. Digesting virtual "geek" culture: the summarization of technical internet relay chats. In *Proceedings of ACL 2005*.

Towards Automatic Scoring of Non-Native Spontaneous Speech

Klaus Zechner and **Isaac I. Bejar**
Educational Testing Service
Princeton, NJ, USA
(kzechner,ibejar)@ets.org

Abstract

This paper investigates the feasibility of automated scoring of spoken English proficiency of non-native speakers. Unlike existing automated assessments of spoken English, our data consists of spontaneous spoken responses to complex test items. We perform both a quantitative and a qualitative analysis of these features using two different machine learning approaches. (1) We use support vector machines to produce a score and evaluate it with respect to a mode baseline and to human rater agreement. We find that scoring based on support vector machines yields accuracies approaching inter-rater agreement in some cases. (2) We use classification and regression trees to understand the role of different features and feature classes in the characterization of speaking proficiency by human scorers. Our analysis shows that across all the test items most or all the feature classes are used in the nodes of the trees suggesting that the scores are, appropriately, a combination of multiple components of speaking proficiency. Future research will concentrate on extending the set of features and introducing new feature classes to arrive at a scoring model that comprises additional relevant aspects of speaking proficiency.

1 Introduction

While automated scoring of open-ended written discourse has been approached by several groups recently (Rudner & Gagne, 2001; Shermis & Burstein, 2003), automated scoring of spontaneous spoken language has proven to be more challenging and complex. Spoken language tests are still mostly scored by human raters. However, several systems exist that score different aspects of spoken language; (Bernstein, 1999; C. Cucchiarini, H. Strik, & L. Boves, 1997a; Franco et al., 2000). Our work departs from previous research in that our goal is to study the feasibility of automating scoring for *spontaneous speech*, that is, when the spoken text is not known in advance.

We approach scoring here as the characterization of a speaker's oral proficiency based on features that can be extracted from a spoken response to a well defined test question by means of automatic speech recognition (ASR). We further approach scoring as the construction of a mapping from a set of features to a score scale, in our case five discrete scores from 1 (least proficient) to 5 (most proficient). The set of features and the specific mapping are motivated by the concept of communicative competence (Bachman, 1990; Canale & Swain, 1980; Hymes, 1972). This means that the features in the scoring system we are developing are meant to characterize specific components of communicative competence, such as mastery of pronunciation, fluency, prosodic, lexical, grammatical and pragmatical subskills. The selection of features is guided by an understanding of the nature of speaking proficiency. We rely on the scoring behavior of judges to evaluate the features (section 8) as well as a convenient criterion for evaluating the feasibility of automated scoring based on those features (section 7). That is, the role of human scorers in this context is to provide a standard for system evaluations (see section 7), as well as to validate specific features and feature classes chosen by the authors (section 8). We use support vector machines (SVMs)

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 216–223,
New York, June 2006.

to determine how well the features recover human scores. We collect performance data under three different conditions, where features are either based on actual recognizer output or on forced alignment. (Forced alignment describes a procedure in speech recognition where the recognizer is looking for the most likely path through the Hidden Markov Models given a transcription of the speech file by an experienced transcriber. This helps, e.g., in finding start and end times of words or phonemes.) We then use classification and regression trees (CART) as a means to evaluate the relative importance and salience of our features. When the classification criterion is a human score, as is the case in this study, an inspection of the CART tree can give us insights into the feature preferences a human judge might have in deciding on a score.

The organization of this paper is as follows: first, we discuss related work in spoken language scoring. Next, we introduce the data of our study and the speech recognizer used. In section 5 we describe features we used for this study. Section 6 describes the agreement among raters for this data. Section 7 describes the SVM analysis, section 8 the CART analysis. This is followed by a discussion and then finally by conclusions and an outlook on future work.

2 Related work

There has been previous work to characterize aspects of communicative competence such as fluency, pronunciation, and prosody. (Franco et al., 2000) present a system for automatic evaluation of pronunciation performance on a phone level and a sentence level of native and non-native speakers of English and other languages (EduSpeak). Candidates read English text and a forced alignment between the speech signal and the ideal path through the Hidden Markov Model (HMM) was computed. Next, the log posterior probabilities for pronouncing a certain phone at a certain position in the signal were computed to achieve a local pronunciation score. These scores are then combined with other automatically derived measures such as the rate of speech (number of words per second) or the duration of phonemes to yield global scores.

(C. Cucchiarini, S. Strik, & L. Boves, 1997b)) and (Cucchiarini et al., 1997a)) describe a system for Dutch pronunciation scoring along similar lines. Their feature set, however, is more extensive and contains, in addition to log likelihood Hidden Markov Model scores, various duration scores, and information on pauses, word stress, syllable structure, and intonation. In an evaluation, they find good agreement between human scores and machine scores.

(Bernstein, 1999)) presents a test for spoken English (SET-10) that has the following types of items: reading, repetition, fill-in-the-blank, opposites and open-ended answers. All types except for the last are scored automatically and a score is reported that can be interpreted as an indicator of how native-like a speaker's speech is. In (Bernstein, DeJong, Pisoni, & Townshend, 2000), an experiment is performed to establish the generalizability of the SET-10 test. It is shown that this test's output can successfully be mapped to the Council of Europe's Framework for describing second language proficiency (North, 2000). This paper further reports on studies done to correlate the SET-10 with two other tests of English proficiency, which are scored by humans and where communicative competence is tested for. Correlations were found to be between 0.73 and 0.88.

3 Data

The data we are using for the experiments in this paper comes from a 2002 trial administration of TOEFLiBT® (Test Of English as a Foreign Language—internet-Based Test) for non-native speakers (LanguEdge ™). Item responses were transcribed from the digital recording of each response. In all there are 927 responses from 171 speakers. Of these, 798 recordings were from one of five main test items, identified as P-A, P-C, P-T, P-E and P-W. The remaining 129 responses were from other questions. As reported below, we use all 927 responses in the adaptation of the speech recognizer but the SVM and CART analyses are based on the 798 responses to the five test items. Of the five test items, three are *independent* tasks (P-A, P-C, P-T) where candidates have to talk freely about a certain topic for 60 seconds. An example might be "Tell me about your favorite teacher." Two of

the test items are *integrated* tasks (P-E, P-W) where candidates first read or listen to some material to which they then have to relate in their responses (90 seconds speaking time). An example might be that the candidates listen to a conversational argument about studying at home vs. studying abroad and then are asked to summarize the advantages and disadvantages of both points of view.

The textual transcription of our data set contains about 123,000 words and the audio files are in WAV format and recorded with a sampling rate of 11025Hz and a resolution of 8 bit.

For the purpose of adaptation of the speech recognizer, we split the full data (927 recordings) into a training (596) and a test set (331 recordings). For the CART and SVM analyses we have 511 files in the *train* and 287 files in the *eval* set, summing up to 798. (Both data sets are subsets from the ASR adaptation training and test sets, respectively.) The transcriptions of the audio files were done according to a transcription manual derived from the German VerbMobil project (Burger, 1995). A wide variety of disfluencies are accounted for, such as, e.g., false starts, repetitions, fillers, or incomplete words. One single annotator transcribed the complete corpus; for the purpose of testing inter-coder agreement, a second annotator transcribed about 100 audio files, which were randomly selected from the complete set of 927 files. The disagreement between annotators, measured as word error rate (WER = (substitutions + deletions + insertions) / (substitutions + deletions + correct)) was slightly above 20% (only lexical entries were measured here). This is markedly more disagreement than in other corpora, e.g., in SwitchBoard (Meteer & al., 1995) where disagreements in the order of 5% are reported, but we have non-native speech from speakers at different levels of proficiency which is more challenging to transcribe.

4 Speech recognition system

Our speech recognizer is a gender-independent Hidden Markov Model system that was trained on 200 hours of dictation data by native speakers of English. 32 cepstral coefficients are used; the dictionary has about 30,000 entries. The sampling rate of the recognizer is 16000Hz as opposed to 11025Hz for the LanguEdge™ corpus. The recognizer can accommodate this difference internally by up-sampling the input data stream.

As our speech recognition system was trained on data quite different from our application (dictation vs. spontaneous speech and native vs. non-native speakers) we adapted the system to the LanguEdge ™ corpus. We were able to increase word accuracy on the unseen test set from 15% before adaptation to 33% in the fully adapted model (both acoustic and language model adaptation).

5 Features

Our feature set, partly inspired by (Cucchiarini et al., 1997a), focuses on low-level fluency features, but also includes some features related to lexical sophistication and to content. The feature set also stems, in part, from the written guidelines used by human raters for scoring this data. The features can be categorized as follows: (1) Length measures, (2) lexical sophistication measures, (3) fluency measures, (4) rate measures, and (5) content measures. Table 1 renders a complete list of the features we computed, along with a brief explanation. We do not claim these features to provide a full characterization of communicative competence; they should be seen as a first step in this direction. The goal of the research is to gradually build such a set of features to eventually achieve as large a coverage of communicative competence as possible. The features are computed based on the output of the recognition engine based on either forced alignment or on actual recognition. The output consists of (a) start and end time of every token and hence potential silence in between (used for most features); (b) identity of filler words (for disfluency-related features); and (c) word identity (for content features).

Lexical counts and length measures	
Segdur	Total duration in seconds of all the utterances
Numutt	Number of utterances in the response
Numwds	Total number of word forms in the speech sample
Numdff	Number of disfluencies (fillers)
Numtok	Number of tokens = Numwds+Numdff
Lexical sophistication	
Types	Number of unique word forms in the speech sample
Ttratio	Ratio Types/Numtok (type-token ratio, TTR)
Fluency measures (based on pause information)	
Numsil	Number of silences, excluding silences between utterances
Silpwd	Ratio Numsil/Numwds
Silmean	Mean duration in seconds of all silences in a response to a test item
Silstddv	Standard deviation of silence duration
Rate measures	
Wpsec	Number of words per second
Dpsec.	Number of disfluencies per second
Tpsec	Number of types per second
Silpsec.	Number of silences per second
Content measures	We first compute test-item-specific word vectors with the frequency counts of all words occurring in the *train* set for each test item (wvec_testitem). Then we generate for every item response a word vector in kind (wvec_response) and finally compute the inner product to yield a similarity score: sim = wvec_testitem*wvec_response
Cvfull	wvec_testitem*wvec_response
6 other Cv*-features	As Cvfull but measure similarity to a subset of wvec_testitem, based on the scores in the *train* set (e.g., "all responses with score 1")
Cvlennorm	Length-normalized Cvfull: Cvfull/Numwds

Table 1: List of features with definitions.

6 Inter-rater agreement

The training and scoring procedures followed standard practices in large scale testing. Scorers are trained to apply the scoring standards that have been previously agreed upon by the developers of the test. The training takes the form of discussing multiple instances of responses at each score level. The scoring of the responses used for training other raters is done by more experienced scorers working closely with the designers of the test.

All the 927 speaking samples (see section 3) were rated once by one of several expert raters, which we call Rater1. A second rating was obtained for approximately one half (454) of the speaking samples, which we call Rater2. We computed the exact agreement for all Rater1-Rater2 pairs for all five test items and report the results in the last column of Table 2. Overall, the exact agreement was about 49% and the kappa coefficient 0.34. These are rather low numbers and certainly demonstrate the difficulty of the rating task for humans. Inter-rater agreement for integrated tasks is lower than for independent tasks. We conjecture that this is related to the dual nature of scoring integrated tasks: for one, the communicative competence per se needs to be assessed, but on the other hand so does the correct interpretation of the written or auditory stimulus material. The low agreement in general is also understandable since the number of feature dimensions that have to be mentally inte-

grated pose a significant cognitive load for judges.[1]

7 SVM models

As we have mentioned earlier, the rationale behind using support vector machines for score prediction is to yield a quantitative analysis of how well our features would work in an actual scoring system, measured against human expert raters. The choice of the particular classifier being SVMs was due to their superior performance in many machine learning tasks.

7.1 Support vector machines

Support vector machines (SVMs) were introduced by (Vapnik, 1995) as an instantiation of his approach to model regularization. They attempt to solve a multivariate discrete classification problem where an n-dimensional hyperplane separates the input vectors into, in the simplest case, two distinct classes. The optimal hyperplane is selected to minimize the classification error on the training data, while maintaining a maximally large margin (the distance of any point from the separating hyperplane).

[1] Inter-human agreement rates for written language, such as essays, are significantly higher, around 70-80% with a 5-point scale (Y.Attali, personal communication). More recently we observed agreement rates of about 60% for spoken test items, but here a 4-point scale was used.

7.2 Experiments

We built five SVM models based on the *train* data, one for each of the five test items. Each model has two versions: (a) based on forced alignment with the true reference, representing the case with 100% word accuracy (align), and (b) based on the actual recognition output hypotheses (hypo). The SVM models were tested on the *eval* data set and there were three test conditions: (1) both training and test conditions derived from forced alignment (align-align); (2) models trained on forced alignment and evaluated based on actual recognition hypotheses (align-hypo; this represented the realistic situation that while human transcriptions are made for the training set, they would turn out to be too costly when the system is running continuously); and (3) both training and evaluation are based on ASR output in recognition mode (hypo-hypo).

We identified the best models by running a set of SVMs with varying cost factors, ranging from 0.01 to 15, and three different kernels: radial basis function, and polynomial, of second degree and of third degree. We selected the best performing models measured on the *train* set and report results with these models on the *eval* set. The cost factor for all three configurations varied between 5 and 15 among the five test items, and as best kernel we found the radial basis function in almost all cases, except for some polynomial kernels in the hypo-hypo configuration

	Mode (% of eval set)	Train : align Eval : align	Train : align Eval : hypo	Train : hypo Eval : hypo	Human Rater Agreement (% of all pairs)
P-A (ind)	34	40.7	33.9	35.9	53
P-C (ind)	53	50.0	55.0	56.7	57
P-T (ind)	38	43.4	18.9	37.7	54
P-E (int)	25	42.1	26.3	47.4	43
P-W (int)	29	34.5	20.7	39.7	42

Table 2: Speech scoring: Mode baseline, SVM performance on forced alignment and standard recognition data, and human agreement for all five test items (ind=independent task; int=integrated task).

7.3 Results

Table 2 shows the results for the SVM analysis as well as a baseline measure of agreement and the inter rater agreement. The baseline refers to the expected level of agreement with Rater1 by simply assigning the mode of the distribution of scores for a given question, i.e., to always assign the most frequently occurring score on the *train* set. Table 2 also reports the agreement between trained raters. As can be seen the human agreement is consistently higher than the mode agreement but the difference is less for the integrated questions suggesting that humans scorers found those questions more challenging to score consistently.

The other 3 columns of Table 2 report the results for the perfect agreement between a score assigned by the SVM developed for that test question and Rater1 on the *eval* corpus, which was not used in the development of the SVM. We observe that for the align-align configuration, accuracies are all clearly better than the mode baseline, except for P-C, which has an unusually skewed score distribution and therefore a rather high mode baseline. In the align-hypo case, where SVM models were built based on features derived from ASR forced alignment and where these models were tested using ASR output in recognition mode, we see a general drop in performance – again except for P-C – which is to be expected as the training and test data were derived in different ways. Finally, in the hypo-hypo configuration, using ASR recognition output for both training and testing, SVM models are, in comparison to the align-align models, improved for the two integrated tasks but not for the independent tasks, again except for P-C. The SVM classification accuracies for the integrated tasks are in the range of human scorer agreement, which indicates that a performance ceiling may have been reached already. These results suggest that the recovery of scores is more feasible for integrated rather than independent tasks. However, it is also the case that human scorers had more difficulty with the integrated tasks, as discussed in the previous section.

The fact that the classification performance of the hypo-hypo models is not greatly lower than that of the align-align models, and in some cases even higher ---and that with the relatively low word accuracy of 33%---, leads to our conjecture that this could be due to the majority of features being based on measures which do not require a correct word identity such as measures of rate or pauses.

In a recent study (Xi, Zechner, & Bejar, 2006) with a similar speech corpus we found that while the hypo-hypo models are better than the align-align models when using features related to fluency, the converse is true when using word-based vocabulary features.

8 CART models

8.1 Classification and regression trees

Classification and regression trees (CART trees) were introduced by (Breiman, Friedman, Olshen, & Stone, 1984). The goal of a classification tree is to classify the data such that the data in the terminal or classification nodes is as pure as possible meaning all the cases have the same true classification, in the present case a score provided by a human rater, the variable Rater1 above. At the top of the tree all the data is available and is split into two groups based on a split of one of the features available. Each split is treated in the same manner until no further splits are possible, in which case a terminal node has been reached.

8.2 Tree analysis

For each of the five test items described above we estimated a classification tree using as independent variables the features described in Table 1 and as the dependent variable a human score. The trees were built on the *train* set. Table 3 shows the distribution of features in the CART tree nodes of the five test items (rows) based on feature classes (columns). For P-A, for example, it can be seen that three of the feature classes have a count greater than 0. The last column shows the number of classes appearing in the tree and the number of total features, in parentheses. The P-A tree, for example has six features from three classes. The last row summarizes the number of test items that relied on a feature class and the number of features from

that class across all five test items, in parenthesis. For example, Rate and Length were present in every test item and lexical sophistication was present in all but one test item. The table suggests that across all test items there was good coverage of feature classes but length was especially well represented. This is to be expected with a group heterogeneous in speaking proficiency. The length features often were used to classify students in the lower scores, that is, students who could not manage to speak sufficiently to be responsive to the test item.

9 Discussion

9.1 Speech recognition

We successfully adapted an off-the-shelf speech recognition engine for the purpose of assessing spontaneous speaking proficiency. By acoustic and language model adaptation, we were able to markedly increase our speech recognition engine's word accuracy, from initially 15% to eventually 33%. Although a 33% recognition rate is not high by current standards, the hurdles to higher recognition are significant, including the fact that the recognizer's acoustic model was originally trained on quite different data, and the fact that our data is based on highly accented speech from non-native speakers of English of a range of proficiencies, which are harder to recognize than native speakers.

9.2 SVM and CART models

Our goal in this research has been to develop models for automatically scoring communicative competence in non-native speakers of English. The approach we took is to compute features from ASR output that may eventually serve as indicators of communicative competence. We evaluated those features (a) in quantitative respect by using SVM models for score prediction and (b) in qualitative respect in terms of their roles in assigning scores based on a human criterion by means of CART analyses.

We found in the analysis of the SVM models that despite low word accuracy, with ASR recognition as a basis for training and testing, scores near inter-rater agreement levels can be reached for those items that include a listening or reading passage. When simulating perfect word accuracy (in the align-align configuration), 4 of 5 test items achieve scoring accuracies above the mode baseline. These results are very encouraging in the light that we are continuing to add features to the models on various levels of speech proficiency.

Test item	Length	Lexical sophistication	Fluency	Rate	Content	Total: # classes (# features)
P-A	4	1	0	1	0	3 (6)
P-C	4	0	1	1	1	4 (7)
P-T	2	1	0	1	1	4 (5)
P-E	1	1	2	1	1	5 (6)
P-W	1	2	0	1	0	3 (4)
Total # classes (# features)	5 (12)	4 (5)	2 (3)	5 (5)	3 (3)	19 (28)

Table 3: Distribution of features from the nodes of five CART trees (rows) into feature classes (columns). The "totals" in the last colunmn and row count first the number of classes with at least one feature and then sums the features (in parentheses).

CART trees have the advantage of being inspectable and interpretable (unlike, e.g., neural nets or support vector machines with non-linear kernels). It is easy to trace a path from the root of the tree to any leaf node and record the final decisions made along the way. We looked at the distribution of features in these CART tree nodes (Table 3) and found that all the different categories of features were used by the set of trees. For all 5 test items, most classes occurred in the nodes of the respective CART trees (with a minimum of 3 out of 5 classes).

10 Conclusions and future work

This paper is concerned with explorations into scoring spoken language test items of non-native speakers of English. We demonstrated that an extended feature set comprising features related to length, lexical sophistication, fluency, rate and content could be used to predict human scores in SVM models and to illuminate their distribution into five different classes by means of a CART analysis.

An important step for future work will be to train the acoustic and language models of the speech recognizer directly from our corpus; we are additionally planning to use automatic speaker adaptation and to evaluate its benefits. Furthermore we are aware that, maybe with the exception of the classes related to fluency, rate and length, our feature set is as of yet quite rudimentary and will need significant expansion in order to obtain a broader coverage of communicative competence.

In summary, future work will focus on improving speech recognition, and on significantly extending the feature sets in different categories. The eventual goal is to have a well-balanced multi-component scoring system which can both rate non-native speech as closely as possible according to communicative criteria, as well as provide useful feedback for the language learner.

References

Bachman, L. F. (1990). *Fundamental considerations in language testing*. Oxford: Oxford University Press.

Bernstein, J. (1999). *PhonePass Testing: Structure and Construct*. Menlo Park, CA: Ordinate Corporation.

Bernstein, J., DeJong, J., Pisoni, D., & Townshend, B. (2000). *Two experiments in automatic scoring of spoken language proficiency.* Paper presented at the InSTIL2000, Dundee, Scotland.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees.* Belmont, California: Wadsworth Int. Group.

Burger, S. (1995). *Konventionslexikon zur Transliteration von Spontansprache*. Munich, Germany.

Canale, M., & Swain, M. (1980). Theoretical bases of communicative approaches to second language teaching and testing. *Applied Linguistics, 1*(1), 1-47.

Cucchiarini, C., Strik, H., & Boves, L. (1997a, September). *Using speech recognition technology to assess foreign speakers' pronunciation of Dutch.* Paper presented at the Third international symposium on the acquisition of second language speech: NEW SOUNDS 97, Klagenfurt, Austria.

Cucchiarini, C., Strik, S., & Boves, L. (1997b). *Automatic evaluation of Dutch pronunciation by using speech recognition technology.* Paper presented at the IEEE Automatic Speech Recognition and Understanding Workshop, Santa Barbara, CA.

Franco, H., Abrash, V., Precoda, K., Bratt, H., Rao, R., & Butzberger, J. (2000). *The SRI EduSpeak system: Recognition and pronunciation scoring for language learning.* Paper presented at the InSTiLL-2000 (Intelligent Speech Technology in Language Learning), Dundee, Scotland.

Hymes, D. H. (1972). On communicative competence. In J. B. Pride & J. Holmes (Eds.), *Sociolinguistics: selected readings* (pp. 269-293). Harmondsworth, Middlesex: Penguin.

Meteer, M., & al., e. (1995). *Dysfluency Annotation Stylebook for the Switchboard Corpus.*Unpublished manuscript.

North, B. (2000). *The Development of a Common Framework Scale of Language Proficiency*. New York, NY: Peter Lang.

Rudner, L., & Gagne, P. (2001). An overview of three approaches to scoring written essays by computer. *Practical Assessment, Research & Development, 7*(26).

Shermis, M. D., & Burstein, J. (2003). *Automated essay scoring: A cross-disciplinary perspective*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*: Springer.

Xi, X., Zechner, K., & Bejar, I. (2006, April). *Extracting meaningful speech features to support diagnostic feedback: an ECD approach to automated scoring.* Paper presented at the NCME, San Francisco, CA.

Unsupervised and Semi-supervised Learning of Tone and Pitch Accent

Gina-Anne Levow
University of Chicago
1100 E. 58th St.
Chicago, IL 60637 USA
levow@cs.uchicago.edu

Abstract

Recognition of tone and intonation is essential for speech recognition and language understanding. However, most approaches to this recognition task have relied upon extensive collections of manually tagged data obtained at substantial time and financial cost. In this paper, we explore two approaches to tone learning with substantially reductions in training data. We employ both unsupervised clustering and semi-supervised learning to recognize pitch accent in English and tones in Mandarin Chinese. In unsupervised Mandarin tone clustering experiments, we achieve 57-87% accuracy on materials ranging from broadcast news to clean lab speech. For English pitch accent in broadcast news materials, results reach 78%. In the semi-supervised framework, we achieve Mandarin tone recognition accuracies ranging from 70% for broadcast news speech to 94% for read speech, outperforming both Support Vector Machines (SVMs) trained on only the labeled data and the 25% most common class assignment level. These results indicate that the intrinsic structure of tone and pitch accent acoustics can be exploited to reduce the need for costly labeled training data for tone learning and recognition.

1 Introduction

Tone and intonation play a crucial role across many languages. However, the use and structure of tone varies widely, ranging from lexical tone which determines word identity to pitch accent signalling information status. Here we consider the recognition of lexical tones in Mandarin Chinese syllables and pitch accent in English.

Although intonation is an integral part of language and is requisite for understanding, recognition of tone and pitch accent remains a challenging problem. The majority of current approaches to tone recognition in Mandarin and other East Asian tone languages integrate tone identification with the general task of speech recognition within a Hidden Markov Model framework. In some cases tone recognition is done only implicitly when a word or syllable is constrained jointly by the segmental acoustics and a higher level language model and the word identity determines tone identity. Other strategies build explicit and distinct models for the syllable final region, the vowel and optionally a final nasal, for each tone.

Recent research has demonstrated the importance of contextual and coarticulatory influences on the surface realization of tones.(Xu, 1997; Shen, 1990) The overall shape of the tone or accent can be substantially modified by the local effects of adjacent tone and intonational elements. Furthermore, broad scale phenomena such as topic and phrase structure can affect pitch height, and pitch shape may be variably affected by the presence of boundary tones. These findings have led to explicit modeling of tonal

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 224–231,
New York, June 2006.

context within the HMM framework. In addition to earlier approaches that employed phrase structure (Fujisaki, 1983), several recent approaches to tone recognition in East Asian languages (Wang and Seneff, 2000; Zhou et al., 2004) have incorporated elements of local and broad range contextual influence on tone. Many of these techniques create explicit context-dependent models of the phone, tone, or accent for each context in which they appear, either using the tone sequence for left or right context or using a simplified high-low contrast, as is natural for integration in a Hidden Markov Model speech recognition framework. In pitch accent recognition, recent work by (Hasegawa-Johnson et al., 2004) has integrated pitch accent and boundary tone recognition with speech recognition using prosodically conditioned models within an HMM framework, improving both speech and prosodic recognition.

Since these approaches are integrated with HMM speech recognition models, standard HMM training procedures which rely upon large labeled training sets are used for tone recognition as well. Other tone and pitch accent recognition approaches using other classification frameworks such as support vector machines (Thubthong and Kijsirikul, 2001) and decision trees with boosting and bagging (Sun, 2002) have relied upon large labeled training sets - thousands of instances - for classifier learning. This labelled training data is costly to construct, both in terms of time and money, with estimates for some intonation annotation tasks reaching tens of times real-time. This annotation bottleneck as well as a theoretical interest in the learning of tone motivates the use of unsupervised or semi-supervised approaches to tone recognition whereby the reliance on this often scarce resource can be reduced.

Little research has been done in the application of unsupervised and semi-supervised techniques for tone and pitch accent recognition. Some preliminary work by (Gauthier et al., 2005) employs self-organizing maps and measures of f0 velocity for tone learning. In this paper we explore the use of spectral and standard k-means clustering for unsupervised acquisition of tone, and the framework of manifold regularization for semi-supervised tone learning. We find that in clean read speech, unsupervised techniques can identify the underlying Mandarin tone categories with high accuracy, while even on noisier broadcast news speech, Mandarin tones can be recognized well above chance levels, with English pitch accent recognition at near the levels achieved with fully supervised Support Vector Machine (SVM) classifiers. Likewise in the semi-supervised framework, tone classification outperforms both most common class assignment and a comparable SVM trained on only the same small set of labeled instances, without recourse to the unlabeled instances.

The remainder of paper is organized as follows. Section 2 describes the data sets on which English pitch accent and Mandarin tone learning are performed and the feature extraction process. Section 3 describes the unsupervised and semi-supervised techniques employed. Sections 4 and 5 describe the experiments and results in unsupervised and semi-supervised frameworks respectively. Section 6 presents conclusions and future work.

2 Data Sets

We consider two corpora: one in English for pitch accent recognition and two in Mandarin for tone recognition. We introduce each briefly below.

2.1 English Corpus

We employ a subset of the Boston Radio News Corpus (Ostendorf et al., 1995), read by female speaker F2B, comprising 40 minutes of news material. The corpus includes pitch accent, phrase and boundary tone annotation in the ToBI framework (Silverman et al., 1992) aligned with manual transcription and syllabification of the materials. Following earlier research (Ostendorf and Ross, 1997; Sun, 2002), we collapse the ToBI pitch accent labels to four classes: unaccented, high, low, and downstepped high for experimentation.

2.2 Mandarin Chinese Tone Data

Mandarin Chinese is a language with lexical tone in which each syllable carries a tone and the meaning of the syllable is jointly determined by the tone and segmental information. Mandarin Chinese has four canonical lexical tones, typically described as follows: 1) high level, 2) mid-rising, 3) low falling-rising, and 4) high falling.[1] The canonical pitch con-

[1]For the experiments in this paper, we exclude the neutral tone, which appears on unstressed syllables, because the clear

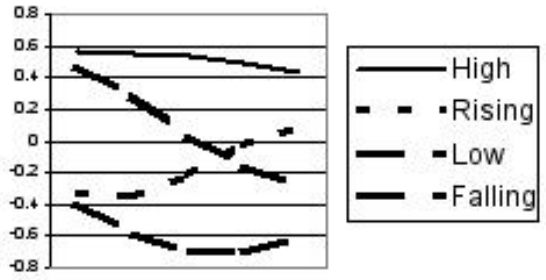


Figure 1: Contours for canonical Mandarin tones

tours for these tones appear in Figure 1.

We employ data from two distinct sources in the experiments reported here.

2.2.1 Read Speech

The first data set is very clean speech data drawn from a collection of read speech collected under laboratory conditions by (Xu, 1999). In these materials, speakers read a set of short sentences where syllable tone and position of focus were varied to assess the effects of focus position on tone realization. Focus here corresponds to narrow focus, where speakers were asked to emphasize a particular word or syllable. Tones on focussed syllables were found to conform closely to the canonical shapes described above, and in previous supervised experiments using a linear support vector machine classifier trained on focused syllables, accuracy approached 99%. For these materials, pitch tracks were manually aligned to the syllable and automatically smoothed and time-normalized by the original researcher, resulting in 20 pitch values for each syllable.

2.2.2 Broadcast News Speech

The second data set is drawn from the Voice of America Mandarin broadcast news, distributed by the Linguistic Data Consortium[2], as part of the Topic Detection and Tracking (TDT-2) evaluation. Using the corresponding anchor scripts, automatically word-segmented, as gold standard transcription, audio from the news stories was force-aligned to the text transcripts. The forced alignment employed the language porting functionality of the University of Colorado Sonic speech recognizer (Pellom et al., 2001). A mapping from the transcriptions to English phone sequences supported by Sonic was created using a Chinese character-pinyin pronunciation dictionary and a manually constructed mapping from pinyin sequences to the closest corresponding English phone sequences.[3]

2.3 Acoustic Features

Using Praat's (Boersma, 2001) "To pitch" and "To intensity" functions and the alignments generated above, we extract acoustic features for the prosodic region of interest. This region corresponds to the "final" region of each syllable in Chinese, including the vowel and any following nasal, and to the syllable nucleus in English.[4] For all pitch and intensity features in both datasets, we compute per-speaker z-score normalized log-scaled values. We extract pitch values from points across valid pitch tracked regions in the syllable. We also compute mean pitch across the syllable. Recent phonetic research (Xu, 1997; Shih and Kochanski, 2000) has identified significant effects of carryover coarticulation from preceding adjacent syllable tones. To minimize these effects consistent with the pitch target approximation model (Xu et al., 1999), we compute slope features based on the second half of this final region, where this model predicts that the underlying pitch height and slope targets of the syllable will be most accurately approached. We further log-scale and normalize slope values to compensate for greater speeds of pitch fall than pitch rise(Xu and Sun, 2002).

We consider two types of contextualized features as well, to model and compensate for coarticulatory effects from neighboring syllables. The first set of features, referred to as "extended features", includes the maximum and mean pitch from adjacent syllables as well as the nearest pitch point or points from the preceding and following syllables. These features extend the modeled tone beyond the strict bounds of the syllable segmentation. A second set of contextual features, termed "difference features", captures the change in pitch maximum, mean, midpoint, and slope as well as intensity maximum be-

speech data described below contains no such instances.

[2]http://www.ldc.upenn.edu

[3]All tone transformations due to third tone sandhi are applied to create the label set.

[4]We restrict our experiments to syllables with at least 50 ms of tracked pitch in this final region.

tween the current syllable and the previous or following syllable.

In prior supervised experiments using support vector machines(Levow, 2005), variants of this representation achieved competitive recognition levels for both tone and pitch accent recognition. Since many of the experiments for Mandarin Chinese tone recognition deal with clean, careful lab speech, we anticipate little coarticulatory influence, and use a simple pitch-only context-free representation for our primary Mandarin tone recognition experiments. For primary experiments in pitch accent recognition, we employ a high-performing contextualized representation in (Levow, 2005), using both "extended" and "difference" features computed only on the preceding syllable. We will also report some contrastive experimental results varying the amount of contextual information.

3 Unsupervised and Semi-supervised Learning

The bottleneck of time and monetary cost associated with manual annotation has generated significant interest in the development of techniques for machine learning and classification that reduce the amount of annotated data required for training. Likewise, learning from unlabeled data aligns with the perspective of language acquisition, as child learners must identify these linguistic categories without explicit instruction by observation of natural language interaction. Of particular interest are techniques in unsupervised and semi-supervised learning where the structure of unlabeled examples may be exploited. Here we consider both unsupervised techniques with no labeled training data and semi-supervised approaches where unlabeled training data is used in conjunction with small amounts of labeled data.

A wide variety of unsupervised clustering techniques have been proposed. In addition to classic clustering techniques such as k-means, recent work has shown good results for many forms of spectral clustering including those by (Shi and Malik, 2000; Belkin and Niyogi, 2002; Fischer and Poland, 2004). In the unsupervised experiments reported here, we employ asymmetric k-lines clustering by (Fischer and Poland, 2004) using code available at the authors' site, as our primary unsupervised learning approach. Asymmetric clustering is distinguished from other techniques by the construction and use of context-dependent kernel radii. Rather than assuming that all clusters are uniform and spherical, this approach enhances clustering effectiveness when clusters may not be spherical and may vary in size and shape. We will see that this flexibility yields a good match to the structure of Mandarin tone data where both shape and size of clusters vary across tones. In additional contrastive experiments reported below, we also compare k-means clustering, symmetric k-lines clustering (Fischer and Poland, 2004), and Laplacian Eigenmaps (Belkin and Niyogi, 2002) with k-lines clustering. The spectral techniques all perform spectral decomposition on some representation of the affinity or adjacency graph.

For semi-supervised learning, we employ learners in the Manifold Regularization framework developed by (Belkin et al., 2004). This work postulates an underlying intrinsic distribution on a low dimensional manifold for data with an observed, ambient distribution that may be in a higher dimensional space. It further aims to preserve locality in that elements that are neighbors in the ambient space should remain "close" in the intrinsic space. A semi-supervised classification algorithm, termed "Laplacian Support Vector Machines", allows training and classification based on both labeled and unlabeled training examples.

We contrast results under both unsupervised and semi-supervised learning with most common class assignment and previous results employing fully supervised approaches, such as SVMs.

4 Unsupervised Clustering Experiments

We executed four sets of experiments in unsupervised clustering using the (Fischer and Poland, 2004) asymmetric clustering algorithm.

4.1 Experiment Configuration

In these experiments, we chose increasingly difficult and natural test materials. In the first experiment with the cleanest data, we used only focused syllables from the read Mandarin speech dataset. In the second, we included both in-focus (focused)

and pre-focus syllables from the read Mandarin speech dataset.[5] In the third and fourth experiments, we chose subsets of broadcast news report data, from the Voice of America (VOA) in Mandarin and Boston University Radio News corpus in English.

In all experiments on Mandarin data, we performed clustering on a balanced sampling set of tones, with 100 instances from each class[6], yielding a baseline for assignment of a single class to all instances of 25%. We then employed a two-stage repeated clustering process, creating 2 or 3 clusters at each stage.

For experiments on English data, we extracted a set of 1000 instances, sampling pitch accent types according to their frequency in the collection. We performed a single clustering phase with 2 to 16 clusters, reporting results at different numbers of clusters.

For evaluation, we report accuracy based on assigning the most frequent class label in each cluster to all members of the cluster.

4.2 Experimental Results

We find that in all cases, accuracy based on the asymmetric clustering is significantly better than most common class assignment and in some cases approaches labelled classification accuracy. Unsurprisingly, the best results, in absolute terms, are achieved on the clean focused syllables, reaching 87% accuracy. For combined in-focus and pre-focus syllables, this rate drops to 77%. These rates contrast with 99-93% accuracies in supervised classification using linear SVM classifiers with several thousand labelled training examples(Surendran et al., 2005).

On broadcast news audio, accuracy for Mandarin reaches 57%, still much better than the 25% level, though below a 72% accuracy achieved using supervised linear SVMs with 600 labeled training examples. Interestingly, for English pitch accent recognition, accuracy reaches 78.4%, aproaching the 80.1% accuracy achieved with SVMs on a comparable data representation.

[5]Post-focus syllables typically have decreased pitch height and range, resulting in particularly poor recognition accuracy. We chose not to concentrate on this specific tone modeling problem here.

[6]Sample sizes were bounded to support rapid repeated experimentation and for consistency with the relatively small VOA data set.

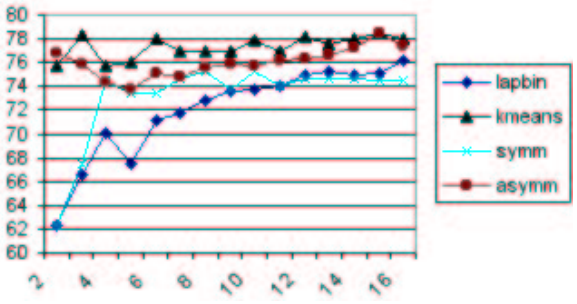


Figure 2: Differences for alternative unsupervised learners across numbers of clusters.

4.3 Contrastive Experiments

We further contrast the use of different unsupervised learners, comparing the three spectral techniques and k-means with Euclidean distance. All contrasts are presented for English pitch accent classification, ranging over different numbers of clusters, with the best parameter setting of neighborhood size. The results are illustrated in Figure 2. K-means and the asymmetric clustering technique are presented for the clean focal Mandarin speech under the standard two stage clustering, in Table 1.

The asymmetric k-lines clustering approach consistently outperforms the corresponding symmetric clustering learner, as well as Laplacian Eigenmaps with binary weights for pitch accent classification. Somewhat surprisingly, k-means clustering outperforms all of the other approaches when producing 3-14 clusters. Accuracy for the optimal choice of clusters and parameters is comparable for asymmetric k-lines clustering and k-means, and somewhat better than all other techniques considered. The careful feature selection process for tone and pitch accent modeling may reduce the difference between the spectral and k-means approaches. In contrast, for the four tone classification task in Mandarin using two stage clustering with 2 or 3 initial clusters, the best clustering using asymmetric k-lines strongly outperforms k-means.

We also performed a contrastive experiment in pitch accent recognition in which we excluded contextual information from both types of contextual features. We find little difference for the majority of

	Asymm.	K-means
Clear speech	87%	74.75%

Table 1: Clustering effectiveness for asymmetric k-lines and k-means on clear focused speech.

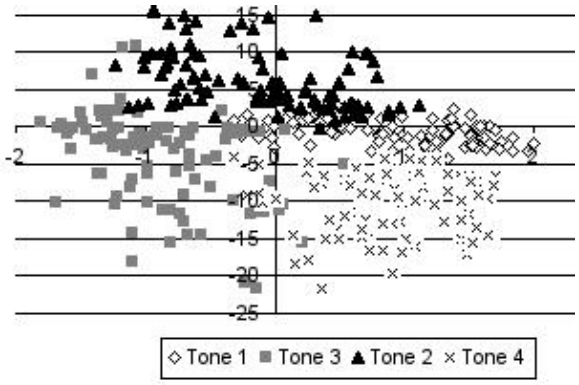


Figure 3: Scatterplot of pitch height vs pitch slope. Open Diamond: High tone (1), Filled black traingle: Rising tone (2), Filled grey square: Low tone (3), X: Falling tone (4)

the unsupervised clustering algorithms, with results from symmetric, asymmetric and k-means clustering differing by less than 1% in absolute accuracy. It is, however, worth noting that exclusion of these features from experiments using supervised learning led to a 4% absolute reduction in accuracy.

4.4 Discussion

An examination of both the clusters formed and the structure of the data provides insight into the effectiveness of this process. Figure 3 displays 2 dimensions of the Mandarin four-tone data from the focused read speech, where normalized pitch mean is on the x-axis and slope is on the y-axis. The separation of classes and their structure is clear. One observes that rising tone (tone 2) lies above the x-axis, while high-level (tone 1) lies along the x-axis. Low (tone 3) and falling (tone 4) tones lie mostly below the x-axis as they generally have falling slope. Low tone (3) appears to the left of falling tone (4) in the figure, corresponding to differences in mean pitch.

In clustering experiments, an initial 2- or 3-way split separates falling from rising or level tones based on pitch slope. The second stage of clustering splits either by slope (tones 1,2, some 3) or by pitch height (tones 3,4). These clusters capture the natural structure of the data where tones are characterized by pitch height and slope targets.

5 Semi-supervised Learning

By exploiting a semi-supervised approach, we hope to enhance classification accuracy over that achievable by unsupervised methods alone by incorporating small amounts of labeled data while exploiting the structure of the unlabeled examples.

5.1 Experiment Configuration

We again conduct contrastive experiments using both the clean focused read speech and the more challenging broadcast news data. In each Mandarin case, for each class, we use only a small set (40) of labeled training instances in conjunction with an additional sixty unlabeled instances, testing on 40 instances. For English pitch accent, we restricted the task to the binary classification of syllables as accented or unaccented. For the one thousand samples we proportionally labeled 200 unaccented examples and 100 accented examples. [7]

We configure the Laplacian SVM classification with binary neighborhood weights, radial basis function kernel, and cosine distance measure typically with 6 nearest neighbors. Following (C-C.Cheng and Lin, 2001), for n-class classification we train $\frac{n(n-1)}{2}$ binary classifiers. We then classify each test instance using all of the classifiers and assign the most frequent prediction, with ties broken randomly. We contrast these results both with conventional SVM classification with a radial basis function kernel excluding the unlabeled training examples and with most common class assignment, which gives a 25% baseline.

5.2 Experimental Results

For the Mandarin focused read syllables, we achieve 94% accuracy on the four-way classification task.

[7] The framework is transductive; the test samples are a subset of the unlabeled training examples.

For the noisier broadcast news data, the accuracy is 70% for the comparable task. These results all substantially outperform the 25% most common class assignment level. The semi-supervised classifier also reliably outperforms an SVM classifier with an RBF kernel trained on the same labeled training instances. This baseline SVM classifier with a very small training set achieves 81% accuracy on clean read speech, but only ≈35% on the broadcast news speech. Finally, for English pitch accent recognition in broadcast news data, the classifier achieves 81.5%, relative to 84% accuracy in the fully supervised case.

6 Conclusion & Future Work

We have demonstrated the effectiveness of both unsupervised and semi-supervised techniques for recognition of Mandarin Chinese syllable tones and English pitch accents using acoustic features alone to capture pitch target height and slope. Although outperformed by fully supervised classification techniques using much larger samples of labelled training data, these unsupervised and semi-supervised techniques perform well above most common class assignment, in the best cases approaching 90% of supervised levels, and, where comparable, well above a good discriminative classifier trained on a comparably small set of labelled data. Unsupervised techniques achieve accuracies of 87% on the cleanest read speech, reaching 57% on data from a standard Mandarin broadcast news corpus, and over 78% on pitch accent classification for English broadcast news. Semi-supervised classification in the Mandarin four-class classification task reaches 94% accuracy on read speech, 70% on broadcast news data, improving dramatically over both the simple baseline of 25% and a standard SVM with an RBF kernel trained only on the labeled examples.

Future work will consider a broader range of tone and intonation classification, including the richer tone set of Cantonese as well as Bantu family tone languages, where annotated data truly is very rare. We also hope to integrate a richer contextual representation of tone and intonation consistent with phonetic theory within this unsupervised and semi-supervised learning framework. We will further explore improvements in classification accuracy based on increases in labeled and unlabeled training examples.

Acknowledgements

We would like to thank Yi Xu for granting access to the read speech data, Vikas Sindhwani, Mikhail Belkin, and Partha Niyogi for their implementation of Laplacian SVM, and Igor Fischer and J. Poland for their implementation of asymmetric clustering.

References

Mikhail Belkin and Partha Niyogi. 2002. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proceeding of NIPS'02*.

M. Belkin, P. Niyogi, and V. Sindhwani. 2004. Manifold regularization: a geometric framework for learning from examples. Technical Report TR-2004-06, University of Chicago Computer Science.

P. Boersma. 2001. Praat, a system for doing phonetics by computer. *Glot International*, 5(9–10):341–345.

C-C.Cheng and C-J. Lin. 2001. LIBSVM:a library for support vector machines. Software available at: http://www.csie.ntu.edu.tw/ cjlin/libsvm.

I. Fischer and J. Poland. 2004. New methods for spectral clustering. Technical Report ISDIA-12-04, IDSIA.

H. Fujisaki. 1983. Dynamic characteristics of voice fundamental frequency in speech and singing. In *The Production of Speech*, pages 39–55. Springer-Verlag.

Bruno Gauthier, Rushen Shi, Yi Xu, and Robert Proulx. 2005. Neural-network simulation of tonal categorization based on f0 velocity profiles. *Journal of the Acoustical Society of America*, 117, Pt. 2:2430.

M. Hasegawa-Johnson, Jennifer Cole, Chilin Shih abd Ken Chen, Aaron Cohen, Sandra Chavarria, Heejin Kim, Taejin Yoon, Sarah Borys, and Jeung-Yoon Choi. 2004. Speech recognition models of the interdependence among syntax, prosody, and segmental acoustics. In *HLT/NAACL-2004*.

Gina-Anne Levow. 2005. Context in multi-lingual tone and pitch accent prediction. In *Proc. of Interspeech 2005 (to appear)*.

M. Ostendorf and K. Ross. 1997. A multi-level model for recognition of intonation labels. In Y. Sagisaka, N. Campbell, and N. Higuchi, editors, *Computing Prosody*, pages 291–308.

M. Ostendorf, P. J. Price, and S. Shattuck-Hufnagel. 1995. The Boston University radio news corpus. Technical Report ECS-95-001, Boston University.

B. Pellom, W. Ward, J. Hansen, K. Hacioglu, J. Zhang, X. Yu, and S. Pradhan. 2001. University of Colorado dialog systems for travel and navigation.

Xiao-Nan Shen. 1990. Tonal co-articulation in Mandarin. *Journal of Phonetics*, 18:281–295.

Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8).

C. Shih and G. P. Kochanski. 2000. Chinese tone modeling with stem-ml. In *Proceedings of the International Conference on Spoken Language Processing, Volume 2*, pages 67–70.

K. Silverman, M. Beckman, J. Pitrelli, M. Ostendorf, C. Wightman, P. Price, J. Pierrehumbert, and J. Hirschberg. 1992. ToBI: A standard for labelling English prosody. In *Proceedings of ICSLP*, pages 867–870.

Xuejing Sun. 2002. Pitch accent prediction using ensemble machine learning. In *Proceedings of ICSLP-2002*.

D. Surendran, Gina-Anne Levow, and Yi Xu. 2005. Tone recognition in Mandarin using focus. In *Proc. of Interspeech 2005 (to appear)*.

Nuttakorn Thubthong and Boonserm Kijsirikul. 2001. Support vector machines for Thai phoneme recognition. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(6):803–813.

C. Wang and S. Seneff. 2000. Improved tone recognition by normalizing for coarticulation and intonation effects. In *Proceedings of 6th International Conference on Spoken Language Processing*.

Yi Xu and X. Sun. 2002. Maximum speed of pitch change and how it may relate to speech. *Journal of the Acoustical Society of America*, 111.

C.X. Xu, Y. Xu, and L.-S. Luo. 1999. A pitch target approximation model for f0 contours in Mandarin. In *Proceedings of the 14th International Congress of Phonetic Sciences*, pages 2359–2362.

Yi Xu. 1997. Contextual tonal variations in Mandarin. *Journal of Phonetics*, 25:62–83.

Y. Xu. 1999. Effects of tone and focus on the formation and alignment of f0 contours - evidence from Mandarin. *Journal of Phonetics*, 27.

J. L. Zhou, Ye Tian, Yu Shi, Chao Huang, and Eric Chang. 2004. Tone articulation modeling for Mandarin spontaneous speech recognition. In *Proceedings of ICASSP 2004*.

Learning Pronunciation Dictionaries
Language Complexity and Word Selection Strategies

John Kominek Alan W Black
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{jkominek,awb}@cs.cmu.edu

Abstract

The speed with which pronunciation dictionaries can be bootstrapped depends on the efficiency of learning algorithms and on the ordering of words presented to the user. This paper presents an active-learning word selection strategy that is mindful of human limitations. Learning rates approach that of an oracle system that knows the final LTS rule set.

1 Introduction

The construction of speech-to-speech translation systems is difficult, complex, and prohibitively expensive for all but handful of major languages. Developing systems for new languages is a highly skilled job requiring considerable effort, as is the process of training people to acquire the necessary technical knowledge.

Ideally, a native speaker of a (minor) language – with the right tools – should be able to develop a speech system with little or no technical knowledge of speech recognition, machine translation, dialog management, or speech synthesis. Rapid development of machine translation, for example, is the goal of (Lavie *et al*., 2003). Similarly, combined development of speech recognition and speech synthesis is the stated goal of (Engelbrecht and Schultz, 2005).

Here we concentrate on lexicon creation for synthesis and recognition tasks, with the affiliated problem of letter-to-sound rule inference. Two central questions of dictionary building are: what letter-to-sound rule representation lends itself well to incremental learning? – and which words should be presented to the user, in what order?

In this paper we investigate various approaches to the word ordering problem, including an active learning algorithm. An "active learner" is a class of machine learning algorithms that choose the order in which it is exposed to training examples (Auer, 2000). This is valuable when there isn't a pre-existing set of training data and when the cost of acquiring such data is high. When humans are adding dictionary entries the time and accuracy depends on the selected word (short words are easier than long; familiar are easier than unfamiliar), and on how quickly the learner's error rate drops (long words are more informative than short). Also, mindful that no answer key exists for new languages – and that humans easily become impatient – we would like to know when a language's letter to sound rule system is, say, 90% complete. This turns out to be surprising elusive to pin down.

The next section outlines our working assumptions and issues we seek to address. Section 3 describes our LTS learning framework, an elaboration of (Davel and Barnard, 2003). The learning behavior on multiple test languages is documented in Section 4, followed in Section 5 by a comparison of several word selection strategies.

2 Assumptions and Issues

In designing language technology development tools we find it helpful to envision our target user, whom may be characterized as "non-technical." Such a person speaks, reads, and writes the target language, is able to enumerate the character set of that language, distinguish punctuation from whitespace, numerals, and regular letters or graphemes, and specify if the language distinguishes upper and lower casing. When presented

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 232–239,
New York, June 2006.

with the pronunciation of a word (as a synthesized wavefile), the user can say whether it is right or wrong. In addition, such a person has basic computer fluency, can record sound files, and can navigate the HTML interface of our software tools. If these latter requirements present a barrier then we assume the availability of a field agent to configure the computer, familiarize the user, plus translate the English instructions, if necessary.

Ideally, our target user need not have explicit knowledge of their own language's phoneme set, nor even be aware that a word can be transcribed as a sequence of phonemes (differently from letters). The ability to reliably discover a workable phoneme set from an unlabeled corpus of speech is not yet at hand, however. Instead we elicit a language's phoneme set during an initialization stage by presenting examples of IPA wavefiles (Wells and House, 1995).

Currently, pronunciations are spelled out using a romanized phonetic alphabet. Following the recommendation of (Davel and Barnard, 2005) a candidate pronunciation is accompanied with a wavefile generated from a phoneme-concatenation synthesizer. Where possible, more than one pronunciation is generated for each word presented, under that assumption that it is easier for a listener to select from among a small number of choices than correct a wrong prediction.

2.1 Four Questions to Address

1. *What is our measure of success?* Ultimately, the time to build a lexicon of a certain coverage and correctness. As a proxy for time we use the number of characters presented. (Not words, as is typically the case, since long words contain more information than short, and yet are harder for a human to verify.)
2. *For a given language, how many words (letters) are needed to learn its LTS rule system?* The true, yet not too useful answer is "it depends." The complexity of the relation between graphemic representation and acoustic realization varies greatly across languages. That being the case, we seek a useful measure of a language's degree of complexity.
3. *Can the asymptote of the LTS system be estimated, so that one can determine when the learned rules are 90 or 95% complete?* In Section 4 we present evidence that this may not be possible. The fall-back position is percentage coverage of the supplied corpus.
4. *Which words should be presented to the user, and in what order?* Each additional word should maximize the marginal information gain to the system. However, short words are easier for humans to contend with than long. Thus a length-based weighting needs to be considered.

3 LTS Algorithm Basics

A wide variety of approaches have been applied to the problem of letter-to-sound rule induction. Due to simplicity of representation and ease of manipulation, our LTS rule learner follows the Default & Refine algorithm of Davel (Davel and Barnard, 2004). In this framework, each letter *c* is assigned a default production p_1-p_2... denoting the sequence of zero or more phonemes most often associated with that letter. Any exceptions to a letter's default rule is explained in terms of the surrounding context of letters. The default rules have a context width of one (the letter itself), while each additional letter increases the width of the context window. For example, if we are considering the first occurrence of 's' in the word *basics,* the context windows are as listed in Table 1. By convention, the underscore character denotes the predicted position, while the hash represents word termination.

width	*context sets ordered by increasing width*
1	{_}
2	{a_ , _i}
3	{ba_ , a_i , _ic}
4	(#ba_ , ba_i , a_ic , _ics}
5	{#ba_i , ba_ic , a_ics , _ics#}
6	{#ba_ic , ba_ics , a_ics#}
7	{#ba_ics , ba_ics#}
8	{#ba_ics#}

Table 1. Letter contexts for the first 's' in *basics*.

In this position there are 20 possible explanatory contexts. The order in which they are visited defines an algorithm's search strategy. In the class of algorithms knows as "dynamically expanding context (DEC)", contexts are considered top-down as depicted in Table 1. Within one row, some algorithms follow a fixed order (e.g. center, left, right). Another variant tallies the instances of productions

associated with a candidate context and chooses the one with the largest count. For example, in Spanish the letter 'c' may generate K (65%), or TH when followed by e or i (32%), or CH when followed by h (3%). These are organized by frequency into a "rule chain."

Rule rank	*RHS*	*Context*	*Frequency*
1	K	_	65.1%
2	TH	_i	23.6%
3	TH	_e	8.5%
4	CH	_h	2.8%

If desired, rules 2 and 3 in this example can be condensed into 'c' → TH /_{i,e}, but in general are left separated for sake of simplicity.

In our variant, before adding a new rule all possible contexts of all lengths are considered when selecting the best one. Thus the rule chains do not obey a strict order of expanding windows, though shorter contexts generally precede longer ones in the rule chains.

One limitation of our representation is that it does not support gaps in the letter context. Consider the word pairs tom/tome, top/tope, tot/tote. A CART tree can represent this pattern with the rule: *if* (c_{-1} = 't' *and* c_0='o' *and* c_2='e') *then ph=OW*. In practice, the inability to skip letters is not a handicap.

3.1 Multiple Pronunciation Predictions

Given a word, finding the predicted pronunciation is easy. Rule chains are indexed by the letter to be predicted, and possible contexts are scanned starting from the most specific until a match is found. Continuing our example, the first letter in the Spanish word *ciento* fails rule 4, fails rule 3, then matches rule 2 to yield TH. For additional pronunciations the search continues until another match is found: here, the default rule 'c' → K /_. This procedure is akin to predicting from progressively smoother models. In a complex language such as English, a ten letter word can readily generate dozens of alternate pronunciations, necessitating an ordering policy to keep the total manageable.

4 Language Characterization

English is notorious for having a highly irregular spelling system. Conversely, Spanish is admired for its simplicity. Most others lie somewhere in between. To estimate how many words need to be seen in order to acquire 90% coverage of a language's LTS rules, it helps to have a quantitative measure. In this section we offer a perplexity-based measure of LTS regularity and present measurements of several languages with varying corpus size. These measurements establish, surprisingly, that a rule system's perplexity increases without bound as the number of training words increases. This holds true whether the language is simple or complex. In response, we resort to a heuristic measure for positioning languages on a scale of relative difficulty.

4.1 A Test Suite of Seven Languages

Our test suite consists of pronunciation dictionaries from seven languages, with English considered under two manifestations.

English. Version 0.6d of CMU-DICT, considered without stress (39 phones) and with two level stress marking (58 phones). ***German***. The Celex dictionary of 321k entries (Burnage, 1990). ***Dutch***. The Fonilex dictionary of 218k entries (Mertens and Vercammen, 1998). Fonilex defines an abstract phonological level from which specific dialects are specified. We tested on the "standard" dialect. ***Afrikaans***. A 37k dictionary developed locally. Afrikaans is a language of South Africa and is a recent derivative of Dutch. ***Italian***. A 410k dictionary distributed as part of a free Festival-based Italian synthesizer (Cosi, 2000). ***Spanish***. Generated by applying a set of hand written rules to a 52k lexicon. The LTS rules are a part of the standard Festival Spanish distribution. ***Telugu***. An 8k locally developed dictionary. In its native orthography, this language of India possess a highly regular syllabic writing system. We've adopted a version of the Itrans-3 transliteration scheme (Kishore 2003) in which sequences of two to four English letters map onto Telugu phonemes.

4.2 Perplexity as a Measure of Difficulty

A useful way of considering letter to sound production is as a Markov process in which the generator passes through a sequence of states (letters), each probabilistically emitting observation symbols (phonemes) before transitioning to the next state (following letter). For a letter c, the unpredictability of phoneme emission is its entropy $H(c)=-\sum(p_i \log p_i)$ or equivalently its perplexity $P(c)=e^{H(c)}$. The perplexity can be interpreted as

the average number of output symbols generated by a letter. The production perplexity of the character set is the sum of each individual letter's perplexity weighted by its unigram probability p_c.

$$Per_{ave} = \sum_c p_c e^{-\sum_i p_i \log p_i} \quad (1)$$

Continuing with our Spanish example, the letter 'c' emits the observation symbols (K, TH, CH) with a probability distribution of (.651, .321, .028), for a perplexity of 2.105. This computation applies when each letter is assigned a single probabilistic state. The process of LTS rule discovery effectively splits the state 'c' into four context-defined substates: (-,c,-), (-,c,i), (-,c,e), (-,c,h). Each of these states emits only a single symbol. Rule addition is therefore an entropy reduction process; when the rule set is complete the letter-to-sound system has a perplexity of 1, i.e. it is perfectly predictable.

The "price paid" for perfect predictability is a complex set of rule chains. To measure rule complexity we again associate a single state with each letter. But, instead of phonemes, the *rules* are the emission symbols. Thus the letter 'c' emits the symbols (K/_, TH/_i, TH/_e, CH/_h) with a distribution of (.651, .236, .085, .028), for a perplexity of 2.534. Applying equation (1) to the full set of rules defines the LTS system's average perplexity.

4.3 Empirical Measurements

In the Default & Refine representation, the rule chain for each letter is is initialized with its most probably production. Additional context-dependent rules are appended to cover additional letter productions, with the rule offering the greatest incremental coverage being added first. (Ties are broken in an implementation-dependent way.)

Figure 1 uses Spanish to illustrate a characteristic pattern: the increase in coverage as rules are added one at a time. Since the figure of merit is letter-based, the upper curve (% letters correct) increases monotonically, while the middle curve (% words correct) can plateau or decrease briefly.

In the lower curve of Figure 1 the growth procedure is constrained such that all width 1 rules are added before width 2 rules, which in turn must be exhausted before width 3 rules are considered. This constraint leads to its distinctive scalloped shape. The upper limit of the W=1 region shows the performance of the unaided default rules (68% words correct).

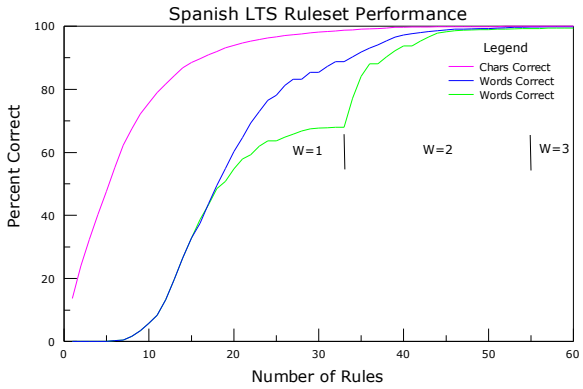


Figure 1. Coverage of Spanish (52k corpus) as a function of rule size. For the lower curve, W indicates the rule context window width. The middle (blue) curve tracks near-optimal performance improvement with the introduction of new rules.

For more complex languages the majority of rules have a context width in the range of 3 to 6. This is seen in Figure 2 for English, Dutch, Afrikaans, and Italian. However, a larger rule set does not mean that the average context width is greater. In Table 2, below, compare Italian to Dutch.

Language	*Number of Rules*	*Average Width*
English 40k	19231	5.06
Dutch 40k	10071	4.35
Afrikaans 37k	5993	4.66
Italian 40k	3385	4.78
Spanish 52k	76	1.66

Table 2. Number of LTS rules for five language and their average context width.

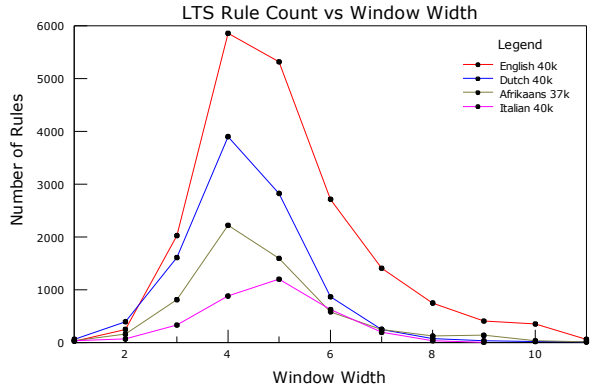


Figure 2. Distribution of LTS rules by context window width for four languages: English, Dutch, Afrikaans, and Italian.

Beyond a window width of 7, rule growth tapers off considerably. In this region most new rules serve to identify particular words of irregular spelling, as it is uncommon for long rules to generalize beyond a single instance. Thus when training a smoothed LTS rule system it is fair to ignore contexts larger than 7, as is done for example in the Festival synthesis system (Black, 1998).

Figure 2 contrasts four languages with training data of around 40k words, but says nothing of how rule sets grow as the corpus size increases. Figure 3 summarizes measurements taken on eight encodings of seven languages (English twice, with and without stress marking), tested from a range of 100 words to over 100,000. Words were subsampled from each alphabetized lexicon at equal spacings. The results are interesting, and for us, unexpected.

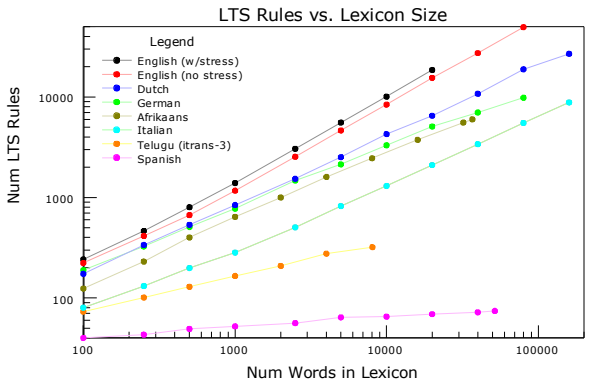


Figure 3. Rule system growth as the corpus size is increased, for seven languages. From top to bottom: English (twice), Dutch, German, Afrikaans, Italian, Telugu, Spanish. The Telugu lexicon uses an Itrans-3 encoding into roman characters, not the native script, which is a nearly perfect syllabic transcription. The context window has a maximum width of 9 in these experiments.

Within this experimental range none of the languages reach an asymptotic limit, though some hint at slowed growth near the upper end. A straight line on a log-log graph is characteristic of geometric growth, to which a power law function $y=ax^b+c$ is an appropriate parametric fit. For difficult languages the growth rates (power exponent b) vary between 0.5 and 0.9, as summarized in Table 3. The language with the fastest growth is English, followed, not by Dutch, but Italian. Italian is nonetheless the simpler of these two, as indicated by the smaller multiplicative factor a.

Language	***a***	***b***
English (stressed)	2.97	0.88
English (plain)	3.27	0.85
Dutch	12.6	0.64
German	39.86	0.49
Afrikaans	15.34	0.57
Italian	2.16	0.69

Table 3. Parameters a and b for the power law fit $y=ax^b+c$ to the growth of LTS system size.

It would be good if a tight ceiling could be estimated from partial data in order to know (and report to the lexicon builder) that with n rules defined the system is m percent complete. However, this trend of geometric growth suggests that asking "how many letter-to-sound rules does a given language have?" is an ill-posed question.

In light of this, two questions are worth asking. First, is the geometric trend particular to our rule representation? And second, is "total number of rules" the right measure of LTS complexity? To answer the first question we repeated the experiments with the CART tree builder available from the Festival speech synthesis toolkit. As it turns out – see Table 4 – a comparison of contextual rules and node counts for Italian demonstrate that a CART tree representation also exhibits geometric growth with respect to lexicon size.

Num Words in Lexicon	***Contextual LTS Rules***	***CART Tree Nodes***
100	80	145
250	131	272
500	198	399
1000	283	601
2500	506	1169
5000	821	1888
10,000	1306	2840
20,000	2109	4642
40,000	3385	7582
80,000	5524	13206

Table 4. A comparison of rule system growth for Italian as the corpus size is increased. CART tree nodes (i.e. questions) are the element comparable to LTS rules used in letter context chains. The fitted parameters to the CART data are a=2.29 and b=0.765. This compares to a=2.16 and b=0.69.

If geometric growth and lack of an obvious asymptote is not particular to expanding context rule chains, then what of the measure? The measure proposed in Section 4.2 is average chain perplexity. The hypothesis is that a system close to saturation will still add new rules, but that the average perplexity levels off. Instead, the data shows little sign of saturation (Figure 4). In contrast, the average perplexity of the letter-to-phoneme distributions remains level with corpus size (Figure 5).

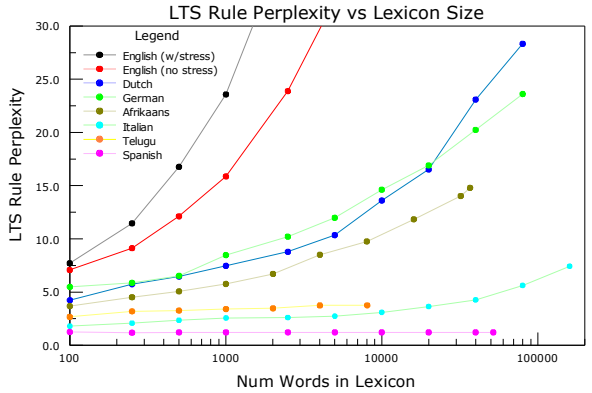


Figure 4. Growth of average rule perplexity as a function of lexicon size. Except for Spanish and Telugu, the average rule system perplexity not only grows, but grows at an accelerating rate.

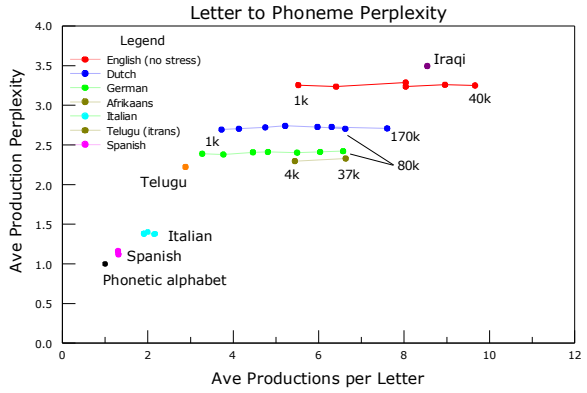


Figure 5. Growth of average letter-to-phoneme production perplexity as a function of lexicon size.

Considering these observations we've resorted to the following heuristic to measure language complexity: a) fix the window width to 5, b) measure the average rule perplexity at lexicon sizes of 10k, 20k, and 40k, then c) take the average of these three values. Fixing the window width to 5 is somewhat arbitrary, but is intended to prevent the system from learning an unbounded suite of exceptions. Available values are contained in Table 5.

Language	***Ave Letter Perplexity***	***Heuristic Perplexity***	***Perplexity Ratio***
English	3.25	50.11	15.42
Dutch	2.73	16.80	6.15
German	2.41	16.70	6.93
Afrikaans	2.32	11.48	8.32
Italian	1.38	3.52	2.55
Spanish	1.16	1.21	1.04

Table 5. Perplexity measures for six languages. The third (rightmost) column is the ratio of the second divided by the first. A purely phonetic system has a heuristic perplexity of one.

From these measurements we conclude, for example, that Dutch and German are equally difficult, that English is 3 times more complex than either of these, and that English is 40 times more complex than Spanish.

5 Word Selection Strategies

A selection strategy is a method for choosing an ordered list of words from a lexicon. It may be based on an estimate of expected maximum return, or be as simple as random selection. A good strategy should enable rapid learning, avoid repetition, be robust, and not overtax the human verifier.

This section compares competing selection strategies on a single lexicon. We've chosen a 10k Italian lexicon as a problem of intermediate difficulty, and focus on early stage learning. To provide a useful frame of reference, Figure 6 shows the results of running 5000 experiments in which the word sequence has been chosen randomly. The x-axis is number of letters examined.

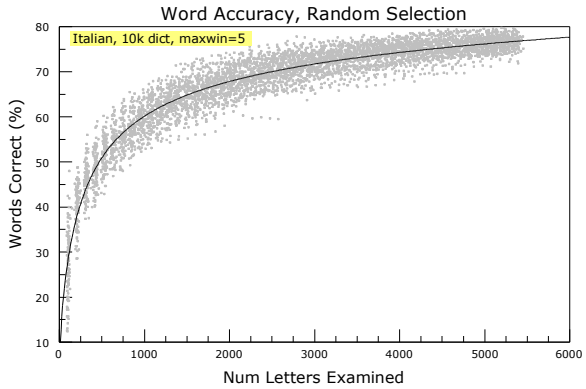


Figure 6. Random sampling of Italian 10k corpus.

Figure 7 compares average random performance to four deterministic strategies. They are: alphabetical word ordering, reverse alphabetical, alphabetical sorted by word length (groups of single character words first, followed by two character words, etc.), and a greedy ngram search. Of the first three, reverse alphabetical performs best because it introduces a greater variety of ngrams more quickly than the others. Yet, all of these three are substantially worse than random. Notice that grouping words from short to long degrades performance. This implies that strategies tuned to the needs of humans will incur a machine learning penalty.

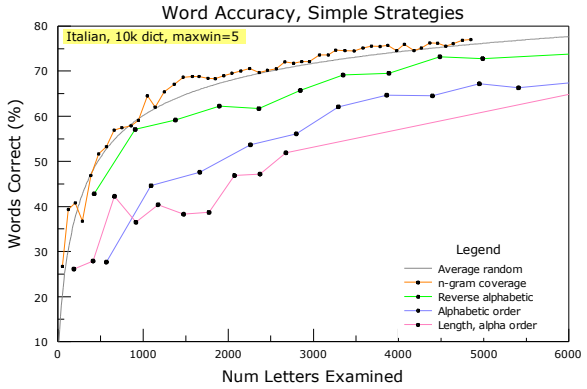


Figure 7. Comparison of three simple word orderings to the average random curve, as well as greedy ngram search.

It might be expected that selecting words containing the most popular ngrams first would out-performs random, but as is seen in Figure 7, greedy selection closely tracks the random curve. This leads us to investigate active leaning algorithms, which we treat as variants of ngram selection.

5.1 Algorithm Description

Let W = $\{w_1,w_2,...\}$ be the lexicon word set, having A = {'a', 'b',...} as the alphabet of letters. We seek an ordered list V = (... w_i ...) s.t. score(w_i) ≥ score (w_{i+1}). V is initially empty and is extended one word at a time with w_b, the "best" new word. Let $g=c_1c_2...c_n \in A^*$ be an ngram of length n, and $G_w=\{g_i\}$, $g_i \in w$ are all the ngrams found in word w. Then $G_W = \uplus G_w$, $w \in W$, is the set of all ngrams in the lexicon W, and $G_V = \uplus G_w$, $w \in V$ is the set of all ngrams in the selected word list V. The number of occurrences of g in W is score(g), while score(w) = ∑ score(g) st. $g \in w$ and $g \notin G_V$. The scored ngrams are segmented into separately sorted lists, forming an ordered list of queues Q = $(q_1,q_2,...q_N)$ where q_n contains ngram of length n and only n.

Algorithm

```
for q in Q
    g = pop(q)
    for L = 1 to |longest word in W|
        W_{g,L} = {w_i} s.t. |w_i| = L, g ∈ w_i and w_i ∉ V
        w_b = argmax score(W_{g,L})
        if score (w_b) > 0 then
            V = V + w_b
            G_V = G_V ∪ G_wb
            return w_b
```

In this search the outer loop orders ngrams by length, while the inner loop orders words by length. For selection based on ngram coverage, the queue Q is computed only once for the given lexicon W. In our active learner, Q is re-evaluated after each word is selected, based on the ngrams present in the current LTS rule contexts. Let $G_{LTS} = \{g_i\}$ s.t. $g_i \in$ some letter context in the LTS rules. Initially $G_{LTS,0} = \{\}$. Then, at any iteration k, $G_{LTS,k}$ are the ngrams present in the rules, and $G'_{LTS,k+1}$ is an expanded set of candidate ngrams that constitute the elements of Q. G' is formed by prepending each letter c of A to each g in G, plus appending each c to g. That is, $G'_{LTS,k+1} = A \times G_{LTS,k} \cup G_{LTS,k} \times A$ where × is the Cartesian product. Executing the algorithm returns w_b and yields $G_{LTS,k+1}$ the set of ngrams covered by the expanded rule set. In this way knowledge of the current LTS rules guides the search for maximally informative new words.

5.2 Active Learner Performance

Figure 8 displays the performance of our active learner on the Italian 10k corpus, shown as the blue curve. For the first 500 characters encountered, the active learner's performance is almost everywhere better than average random, typically one half to one standard deviation above this reference level.

Two other references are shown. Immediately above the active learner curve is "Oracle" word selection. The Oracle has access to the final LTS system and selects words that maximally increases coverage of the known rules. The topmost curve is for a "Perfect Oracle." This represents an even more unrealistic situation in which each letter of each word carries with it information about the corresponding production rule. For example, that 'g' yields /F/ 10% of the time, when followed by the letter 'h' (as in "laugh") . Carrying complete information with each letter allows the LTS system to be constructed directly and without mistake. In contrast, the non-perfect oracle makes mistakes sequencing rules in each letter's rule chain. This decreases performance.

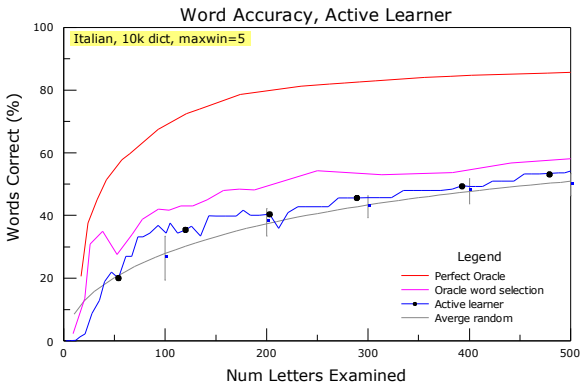


Figure 8. From top to bottom: a perfect Oracle, a word selection Oracle, our active learner, and average random performance. The perfect Oracle demarcates (impossibly high) optimal performance, while Oracle word selection suggests near-optimality. For comparison, standard deviation error bars are added to the random curve.

Encouragingly, the active learning algorithm straddles the range in between average random (the baseline) and Oracle word selection (near-optimality). Less favorable is the non-monotonicity of the performance curve; for example, when the number of letters examined is 135, and 210. Analysis shows that these drops occur when a new letter-to-sound production is encountered but more than one context offers an equally likely explanation. Faced with a tie, the LTS learner sometimes chooses incorrectly. Not being aware of this mistake it does not seek out correcting words. Flat plateaus occur when additional words (containing the next most popular ngrams) do not contain previously unseen letter-to-sound productions.

6 Conclusions

While this work does not definitively answer the question of "how may words to learn the rules," we have developed ways of characterizing language complexity, which can guide developers. We've devised a word selection strategy that appears to perform better than the (surprisingly high) standard set by randomly selection. Further improvements are possible by incorporating knowledge of word alignment and rule sequencing errors. By design, our strategy is biased towards short words over long, thereby being "nice" to lexicon developers – our original objective.

Acknowledgments

This material is in part based upon work supported by the National Science Foundation under Grant No. 0415201. 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 National Science Foundation.

References

Peter Auer, 2000. *Using upper confidence bounds for online learning*. Proceedings of the 41st Annual Symposium on Foundations of Computer Science, pp.

Alan W Black, Kevin Lenzo, and Vincent Pagel, 1998. *Issues in Building General Letter to Sound Rules.* 3rd ESCA Workshop on Speech Synthesis, Australia.

Gavin Burnage, 1990. *CELEX – A Guide for Users*. Hijmegen: Centre for Lexical Information, University of Nijmegen.

Piero Cosi, Roberto Gretter, Fabio Tesser, 2000. *Festival parla italiano*. Proceedings of GFS2000, Giornate del Gruppo di Fonetica Sperimentale, Padova.

Marelie Davel and Etienne Barnard, 2003. *Bootstrapping in Language Resource Generation.* Proceedings of the 14th Symposium of the Pattern Recognition Association of South Africa, pp. 97-100.

Marelie Davel and Etienne Barnard, 2004. *A default-and-refine approach to pronunciation prediction*, Proceedings of the 15th Symposium of the Pattern Recognition Association of South Africa.

Marelie Davel and Etienne Barnard, 2005. *Bootstrapping Pronunciation Dictionaries: Practical Issues.* Proceedings of the 9th International Conference on Spoken Language Processing, Lisbon, Portugal.

Herman Engelbrecht, Tanja Schultz, 2005. *Rapid Development of an Afrikaans-English Speech-to-Speech Translator*, International Workshop on Spoken Language Translation, Pittsburgh, PA. pp.169-176.

S P Kishore and Alan W Black, 2003. *Unit Size in Unit Selection Speech Synthesis.* Proceedings of the 8th European Conference on Spoken Language Processing, Geneva, Switzerland.

Alon Lavie, et al. 2003. *Experiments with a Hindi-to-English Transfer-based MT System under a Miserly Data Scenario*, ACM Transactions on Asian Language Information Processing, 2(2).

Piet Mertens and Filip Vercammen, 1998. *Fonilex Manual*, Technical Report, K. U. Leuven CCL.

John Wells and Jill House, 1995. *Sounds of the IPA*. http://www.phon.ucl.ac.uk/shop/soundsipa.php.

Relabeling Syntax Trees to Improve Syntax-Based Machine Translation Quality

Bryant Huang
Language Weaver, Inc.
4640 Admiralty Way, Suite 1210
Marina del Rey, CA 90292
bhuang@languageweaver.com

Kevin Knight
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292
knight@isi.edu

Abstract

We identify problems with the Penn Treebank that render it imperfect for syntax-based machine translation and propose methods of relabeling the syntax trees to improve translation quality. We develop a system incorporating a handful of relabeling strategies that yields a statistically significant improvement of 2.3 BLEU points over a baseline syntax-based system.

1 Introduction

Recent work in statistical machine translation (MT) has sought to overcome the limitations of phrase-based models (Marcu and Wong, 2002; Koehn et al., 2003; Och and Ney, 2004) by making use of syntactic information. Syntax-based MT offers the potential advantages of enforcing syntax-motivated constraints in translation and capturing long-distance/non-contiguous dependencies. Some approaches have used syntax at the core (Wu, 1997; Alshawi et al., 2000; Yamada and Knight, 2001; Gildea, 2003; Eisner, 2003; Hearne and Way, 2003; Melamed, 2004) while others have integrated syntax into existing phrase-based frameworks (Xia and McCord, 2004; Chiang, 2005; Collins et al., 2005; Quirk et al., 2005).

In this work, we employ a syntax-based model that applies a series of tree/string (xRS) rules (Galley et al., 2004; Graehl and Knight, 2004) to a source language string to produce a target language phrase structure tree. Figure 1 exemplifies the translation process, which is called a *derivation*, from Chinese into English. The source string to translate (枪手 被 警方 击毙 .) is shown at the top left. Rule ① replaces the Chinese word 警方 (shaded) with the English **NP-C** *police*. Rule ② then builds a **VP** over the 被 **NP-C** 击毙 sequence. Next, 枪手 is translated as the **NP-C** *the gunman* by rule ③. Finally, rule ④ combines the sequence of **NP-C VP** . into an **S**, denoting a complete tree. The yield of this tree gives the target translation: *the gunman was killed by police .*

The Penn English Treebank (PTB) (Marcus et al., 1993) is our source of syntactic information, largely due to the availability of reliable parsers. It is not clear, however, whether this resource is suitable, as is, for the task of MT. In this paper, we argue that the overly-general tagset of the PTB is problematic for MT because it fails to capture important grammatical distinctions that are critical in translation. As a solution, we propose methods of relabeling the syntax trees that effectively improve translation quality.

Consider the derivation in Figure 2. The output translation has two salient errors: determiner/noun number disagreement (**this Turkish positions*) and auxiliary/verb tense disagreement (**has demonstrate*). The first problem arises because the **DT** tag, which does not distinguish between singular and plural determiners, allows singular *this* to be used with plural **NNS** *positions*. In the second problem, the **VP-C** tag fails to communicate that it is headed by the base verb (**VB**) *demonstrate*, which should prevent it from being used with the auxiliary **VBZ** *has*. Information-poor tags like **DT** and **VP-C** can be relabeled to encourage more fluent translations, which is the thrust of this paper.

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 240–247,
New York, June 2006.

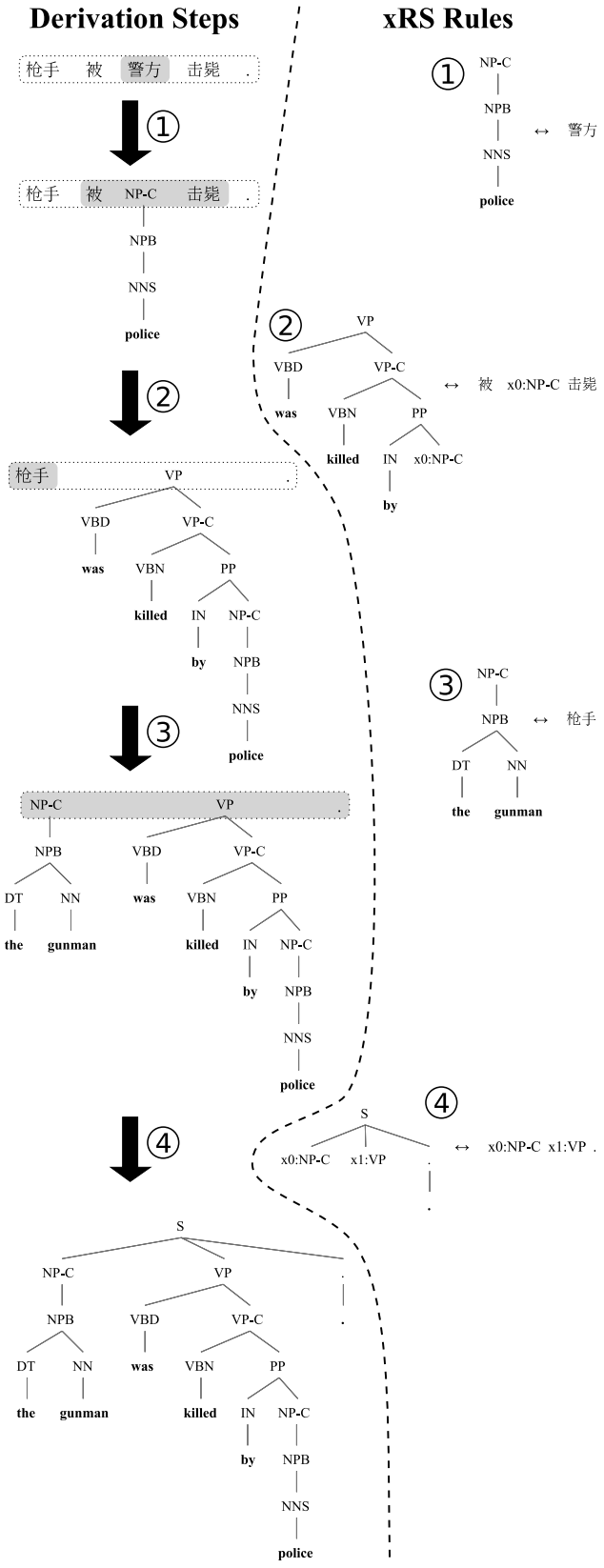


Figure 1: A derivation from a Chinese sentence to an English tree.

Section 2 describes our data and experimental procedure. Section 3 explores different relabeling approaches and their impact on translation quality. Section 4 reports a substantial improvement in BLEU achieved by combining the most effective relabeling methods. Section 5 concludes.

2 Experimental Framework

Our training data consists of 164M+167M words of parallel Chinese/English text. The English half was parsed with a reimplementation of Collins' Model 2 (Collins, 1999) and the two halves were word-aligned using GIZA++ (Och and Ney, 2000). These three components — Chinese strings, English parse trees, and their word alignments — were inputs to our experimental procedure, which involved five steps: (1) tree relabeling, (2) rule extraction, (3) decoding, (4) n-best reranking, (5) evaluation.

This paper focuses on step 1, in which the original English parse trees are transformed by one or more relabeling strategies. Step 2 involves extracting *minimal* xRS rules (Galley et al., 2004) from the set of string/tree/alignments triplets. These rules are then used in a CKY-type parser-decoder to translate the 878-sentence 2002 NIST MT evaluation test set (step 3). In step 4, the output 2,500-sentence n-best list is reranked using an n-gram language model trained on 800M words of English news text. In the final step, we score our translations with 4-gram BLEU (Papineni et al., 2002).

Separately for each relabeling method, we ran these five steps and compared the resulting BLEU score with that of a baseline system with no relabeling. To determine if a BLEU score increase or decrease is meaningful, we calculate statistical significance at 95% using paired bootstrap resampling (Koehn, 2004; Zhang et al., 2004) on 1,000 samples.

Figure 3 shows the results from each relabeling experiment. The second column indicates the change in the number of unique rules from the baseline number of 16.7M rules. The third column gives the BLEU score along with an indication whether it is a statistically significant increase (▲), a statistically significant decrease (▼), or neither (**?**) over the baseline BLEU score.

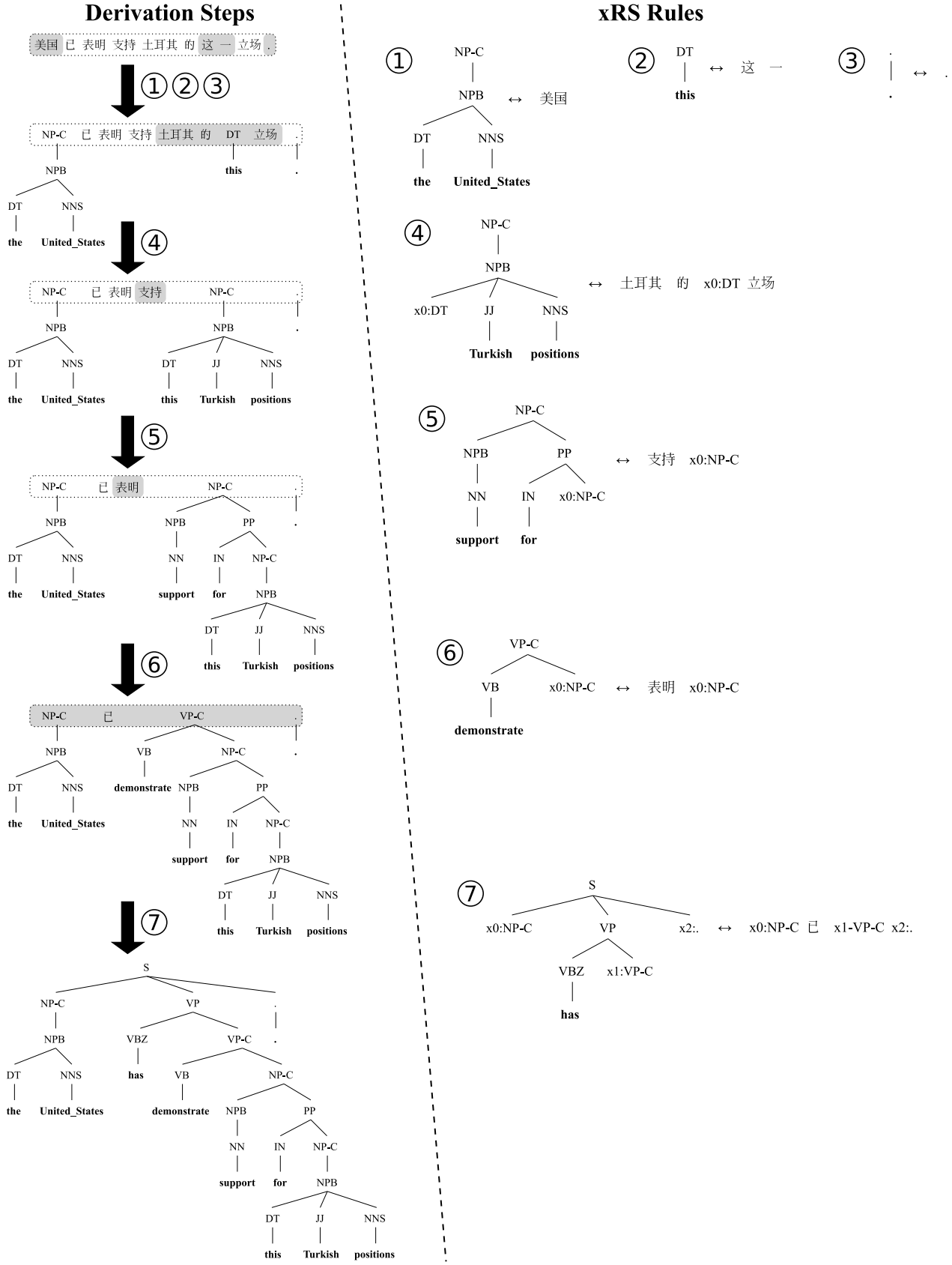


Figure 2: A bad translation fixable by relabeling.

Relabeling	Variant	Δ # Rules	BLEU	Δ
BASELINE		—	20.06	—
LEX_PREP	1	+301.2K	20.2	▲
	2	+254.8K	20.36	▲
	3	+188.3K	20.14	▲
LEX_DT	1	+36.1K	20.15	▲
	2	+29.6K	20.18	▲
LEX_AUX	1	+5.1K	20.09	▲
	2	+8.0K	20.09	?
	3	+1.6K	20.11	▲
	4	+13.8K	20.07	?
LEX_CC		+3.3K	20.03	▼
LEX_%		+0.3K	20.14	▲
TAG_VP		+123.6K	20.28	▲
SISTERHOOD	1	+1.1M	21.33	▲
	2	+935.5K	20.91	▲
	3	+433.1K	20.36	▲
	4	+407.0K	20.59	▲
PARENT	1	+1.1M	19.77	▼
	2	+9.0K	20.01	▼
	3	+2.9M	15.63	▼
COMP_IN		+17.4K	20.36	▲
REM_NPB		–3.5K	19.93	▼
REM_-C		–143.4K	19.3	▼
REM_SG		–9.4K	20.01	▼

Figure 3: For each relabeling method and variant, the impact on ruleset size and BLEU score over the baseline.

3 Relabeling

The small tagset of the PTB has the advantage of being simple to annotate and to parse. On the other hand, this can lead to tags that are overly generic. Klein and Manning (2003) discuss this as a problem in parsing and demonstrate that annotating additional information onto the PTB tags leads to improved parsing performance. We similarly propose methods of relabeling PTB trees that notably improve MT quality. In the next two subsections, we explore relabeling strategies that fall under two categories introduced by Klein and Manning — internal annotation and external annotation.

3.1 Internal Annotation

Internal annotation reveals information about a node and its descendants to its surrounding nodes (ancestors, sisters, and other relatives) that is otherwise hidden. This is paramount in MT because the contents of a node must be understood before the node can be reliably translated and positioned in a sentence. Here we discuss two such strategies: lexicalization and tag annotation.

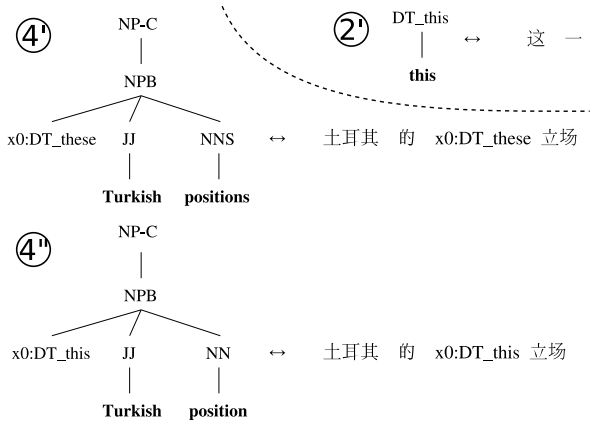


Figure 4: Rules before and after lexicalization.

3.1.1 Lexicalization

Many state-of-the-art statistical parsers incorporate *lexicalization* to effectively capture word-specific behavior, which has proved helpful in our system as well. We generalize lexicalization to allow a lexical item (terminal word) to be annotated onto any ancestor label, not only its parent.

Let us revisit the determiner/noun number disagreement problem in Figure 2 (**this Turkish positions*). If we lexicalize all **DT**s in the parse trees, the problematic **DT** is relabeled more specifically as **DT_this**, as seen in rule ②' in Figure 4. This also produces rules like ④', where both the determiner and the noun are plural (notice the **DT_these**), and ④', where both are singular. With such a ruleset, ②' could only combine with ④', not ④', enforcing the grammatical output *this Turkish position*.

We explored five lexicalization strategies, each targeting a different grammatical category. A common translation mistake was the improper choice of prepositions, e.g., *responsibility **to** attacks*. Lexicalizing prepositions proved to be the most effective lexicalization method (LEX_PREP). We annotated a preposition onto both its parent (**IN** or **TO**) and its grandparent (**PP**) since the generic **PP** tag was often at fault. We tried lexicalizing all prepositions (variant 1), the top 15 most common prepositions (variant 2), and the top 5 most common (variant 3). All gave statistically significant BLEU improvements, especially variant 2.

The second strategy was **DT** lexicalization

(LEX_DT), which we encountered previously in Figure 4. This addresses two features of Chinese that are problematic in translation to English: the infrequent use of articles and the lack of overt number indicators on nouns. We lexicalized these determiners: *the*, *a*, *an*, *this*, *that*, *these*, or *those*, and grouped together those with similar grammatical distributions (*a/an*, *this/that*, and *these/those*). Variant 1 included all the determiners mentioned above and variant 2 was restricted to *the* and *a/an* to focus only on articles. The second slightly improved on the first.

The third type was auxiliary lexicalization (LEX_AUX), in which all forms of the verb *be* are annotated with **_be**, and similarly with *do* and *have*. The PTB purposely eliminated such distinctions; here we seek to recover them. However, auxiliaries and verbs function very differently and thus cannot be treated identically. Klein and Manning (2003) make a similar proposal but omit *do*. Variants 1, 2, and 3, lexicalize *have*, *be*, and *do*, respectively. The third variant slightly outperformed the other variants, including variant 4, which combines all three.

The last two methods are drawn directly from Klein and Manning (2003). In **CC** lexicalization (LEX_CC), both *but* and *&* are lexicalized since these two conjunctions are distributed very differently compared to other conjunctions. Though helpful in parsing, it proved detrimental in our system. In **%** lexicalization (LEX_%), the percent sign (%) is given its own **PCT** tag rather than its typical **NN** tag, which gave a statistically significant BLEU increase.

3.1.2 Tag Annotation

In addition to propagating up a terminal word, we can also propagate up a nonterminal, which we call *tag annotation*. This partitions a grammatical category into more specific subcategories, but not as fine-grained as lexicalization. For example, a **VP** headed by a **VBG** can be tag-annotated as **VP_VBG** to represent a progressive verb phrase.

Let us once again return to Figure 2 to address the auxiliary/verb tense disagreement error (**has demonstrate*). The auxiliary *has* expects a **VP-C**, permitting the bare verb phrase *demonstrate* to be incorrectly used. However, if we tag-annotate all **VP-C**s, rule ⑥ would be relabeled as **VP-C_VB** in rule ⑥′ and rule ⑦ as ⑦′ in Figure 5. Rule ⑥′ can no longer

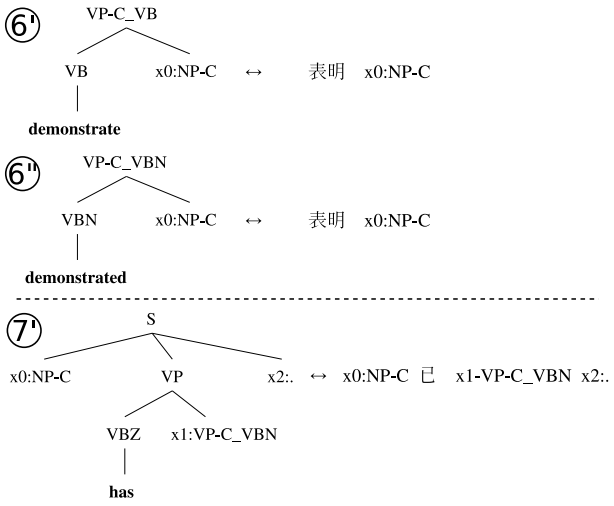


Figure 5: Rules before and after tag annotation.

join with ⑦′, while the variant rule ⑥′ can, which produces the grammatical result *has demonstrated*.

We noticed many wrong verb tense choices, e.g., gerunds and participles used as main sentence verbs. We resolved this by tag-annotating every **VP** and **VP-C** with its head verb (TAG_VP). Note that we group **VBZ** and **VBP** together since they have very similar grammatical distributions and differ only by number. This strategy gave a healthy BLEU improvement.

3.2 External Annotation

In addition to passing information from inside a node to the outside, we can pass information from the external environment into the node through *external annotation*. This allows us to make translation decisions based on the context in which a word or phrase is found. In this subsection, we look at three such methods: sisterhood annotation, parent annotation, and complement annotation.

3.2.1 Sisterhood Annotation

The single most effective relabeling scheme we tried was *sisterhood annotation*. We annotate each nonterminal with **#L** if it has any sisters to the left, **#R** if any to the right, **#LR** if on both sides, and nothing if it has no sisters. This distinguishes between words that tend to fall on the left or right border of a constituent (often head words, like **NN#L** in an **NP** or **IN#R** in a **PP**), in the middle of a constituent (often modifiers, like **JJ#LR** in an **NP**), or by themselves

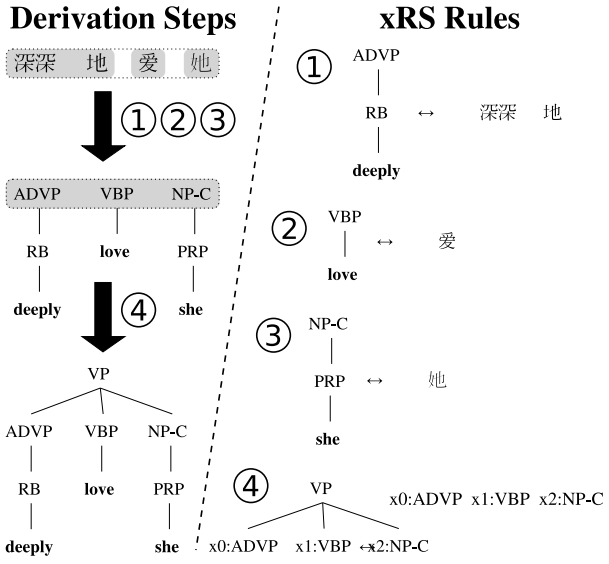


Figure 6: A bad translation fixable by sisterhood or parent annotation.

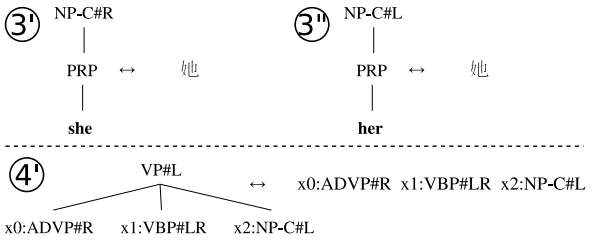


Figure 7: Rules before and after sisterhood annotation.

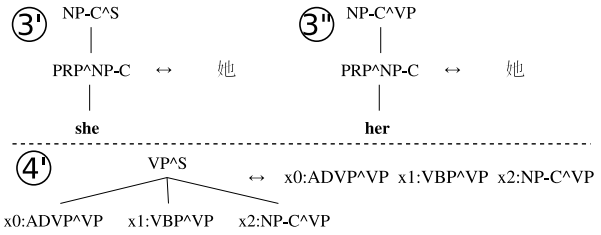


Figure 8: Rules before and after parent annotation.

(often particles and pronouns, like **RP** and **PRP**). In our outputs, we frequently find words used in positions where they should be disallowed or disfavored.

Figure 6 presents a derivation that leads to the ungrammatical output **deeply love she*. The subject pronoun *she* is incorrectly preferred over the object form *her* because the most popular **NP-C** translation for 她 is *she*. We can sidestep this mistake through sisterhood-annotation, which yields the relabeled rules ③' and ④' in Figure 7. Rule ④' expects an **NP-C** on the right border of the constituent (**NP-C#L**). Since *she* never occurs in this position in the PTB, it should never be sisterhood-annotated as an **NP-C#L**. It does occur with sisters to the right, which gives the **NP-C#R** rule ③'. The object **NP-C** *her*, on the other hand, is frequently rightmost in a constituent, which is reflected in the **NP-C#L** rule ③''. Using this rule with rule ④' gives the desired result *deeply love her*.

We experimented with four sisterhood annotation (SISTERHOOD) variants of decreasing complexity. The first was described above, which includes rightmost (**#L**), leftmost (**#R**), middle (**#LR**), and alone (no annotation). Variant 2 omitted **#LR**, variant 3 kept only **#LR**, and variant 4 only annotated nodes without sisters. Variants 1 and 2 produced the largest gains from relabeling: 1.27 and 0.85 BLEU points, respectively.

3.2.2 Parent Annotation

Another common relabeling method in parsing is *parent annotation* (Johnson, 1998), in which a node is annotated with its parent's label. Typically, this is done only to nonterminals, but Klein and Manning (2003) found that annotating preterminals as well was highly effective. It seemed likely that such contextual information could also benefit MT.

Let us tackle the bad output from Figure 6 with parent annotation. In Figure 8, rule ④ is relabeled as rule ④' and expects an **NP-CˆVP**, i.e., an **NP-C** with a **VP** parent. In the PTB, we observe that the **NP-C** *she* never has a **VP** parent, while *her* does. In fact, the most popular parent for the **NP-C** *her* is **VP**, while the most popular parent for *she* is **S**. Rule ③ is relabeled as the **NP-CˆS** rule ③' and *her* is expressed as the **NP-CˆVP** rule ③''. Only rule ③'' can partner with rule ④', which produces the correct output *deeply love her*.

We tested three variants of parent annotation (PARENT): (1) all nonterminals are parent-annotated, (2) only **S** nodes are parent-annotated, and (3) all nonterminals are parent- and grandparent-annotated (the annotation of a node's parent's parent). The first and third variants yielded the largest ruleset sizes of all relabeling methods. The second variant was restricted only to **S** to capture the difference between top-level clauses (**SˆTOP**) and em-

bedded clauses (like **S^S-C**). Unfortunately, all three variants turned out to be harmful in terms of BLEU.

3.2.3 Complement Annotation

In addition to a node's parent, we can also annotate a node's complement. This captures the fact that words have a preference of taking certain complements over others. For instance, 96% of cases where the **IN** *of* takes one complement in the PTB, it takes **NP-C**. On the other hand, *although* never takes **NP-C** but takes **S-C** 99% of the time.

Consider the derivation in Figure 9 that results in the bad output **postponed out May 6*. The **IN** *out* is incorrectly allowed despite the fact that it almost never takes an **NP-C** complement (0.6% of cases in the PTB). A way to restrict this is to annotate the **IN**'s complement. Complement-annotated versions of rules ② and ③ are given in Figure 10. Rule ② is relabeled as the **IN/PP-C** rule ②' since **PP-C** is the most common complement for *out* (99% of the time). Since rule ③' expects an **IN/NP-C**, rule ②' is disqualified. The preposition *from* (rule ②''), on the other hand, frequently takes **NP-C** as complement (82% of the time). Combining rule ②'' with rule ③' ensures the correct output *postponed from May 6*.

Complement-annotating all **IN** tags with their complement if they had one and only one complement (COMP_IN) gave a significant BLEU improvement with only a modest increase in ruleset size.

3.3 Removal of Parser Annotations

Many parsers, though trained on the PTB, do not preserve the original tagset. They may omit function tags (like **-TMP**), indices, and null/gap elements or add annotations to increase parsing accuracy and provide useful grammatical information. It is not obvious whether these modifications are helpful for MT, so we explore the effects of removing them.

The statistical parser we used makes three relabelings: (1) base **NP**s are relabeled as **NPB**, (2) argument nonterminals are suffixed with **-C**, and (3) subjectless sentences are relabeled from **S** to **SG**. We tried removing each annotation individually (REM_NPB, REM_-C, and REM_SG), but doing so significantly dropped the BLEU score. This leads us to conclude these parser additions are helpful in MT.

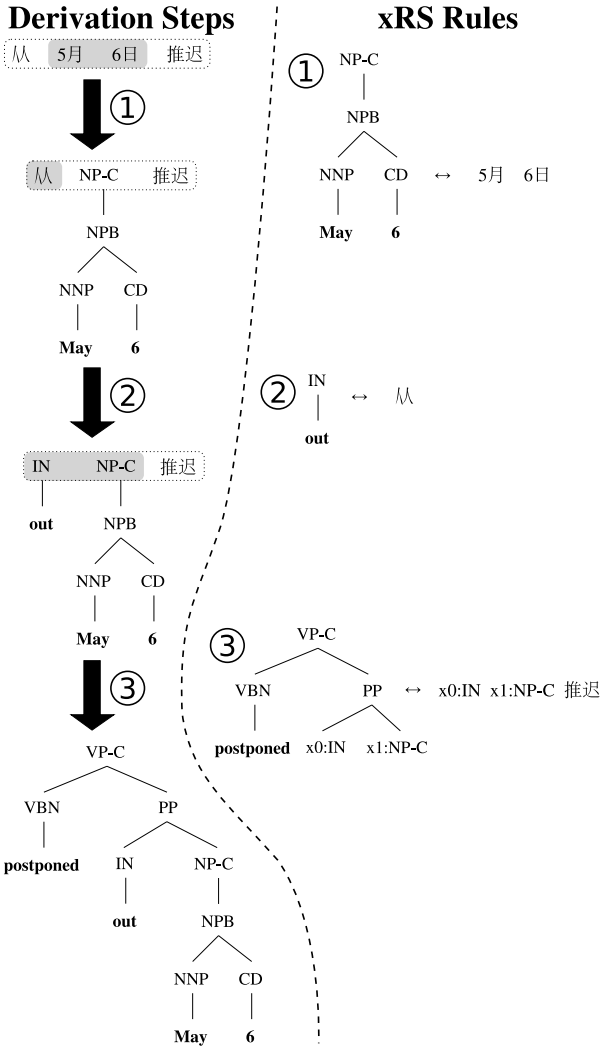


Figure 9: A bad translation fixable by complement annotation.

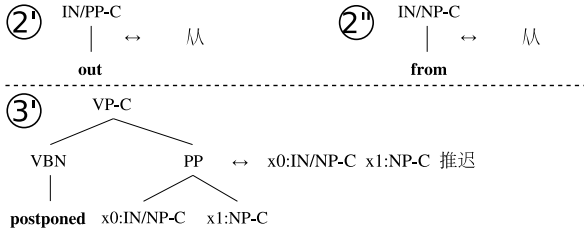


Figure 10: Rules before and after complement annotation.

4 Evaluation

To maximize the benefit of relabeling, we incorporated five of the most promising relabeling strategies into one additive system: LEX_%, LEX_DT variant

		Δ # Rules		BLEU	
Relabeling	**Variant**	**Ind.**	**Cum.**	**Ind.**	**Cum.**
BASELINE		—	—	20.06	20.06
LEX_%		+0.3K	+0.3K	20.14	20.14
LEX_DT	2	+29.6K	+29.9K	20.18	20.3
TAG_VP		+123.6K	+153.5K	20.28	20.43
LEX_PREP	2	+254.8K	+459.0K	20.36	21.25
SISTERHOOD	1	+1.1M	+1.5M	21.33	*22.38*

Figure 11: Relabelings in the additive system and their individual/cumulative effects over the baseline.

2, TAG_VP, LEX_PREP variant 2, and SISTERHOOD variant 1. These relabelings contributed to a 2.3 absolute (11.6% relative) BLEU point increase over the baseline, with a score of 22.38. Figure 11 lists these relabelings in the order they were added.

5 Conclusion

We have demonstrated that relabeling syntax trees for use in syntax-based machine translation can significantly boost translation performance. It is naïve to assume that linguistic resources can be immediately useful out of the box, in our case, the Penn Treebank for MT. Rather, we targeted features of the PTB tagset that impair translatability and proposed relabeling strategies to overcome these weaknesses. Many of our ideas effectively raised the BLEU score over a baseline system without relabeling. Finally, we demonstrated through an additive system that relabelings can be combined together to achieve an even greater improvement in translation quality.

Acknowledgments

This research was supported in part by NSF grant IIS-0428020. We would like to thank Greg Langmead, Daniel Marcu, and Wei Wang for helpful comments. This paper describes work conducted while the first author was at the University of Southern California/Information Sciences Institute.

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.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL-05*.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL-05*, pages 531–540.

Michael Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of ACL-03 (Companion Volume)*, pages 205–208.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT/NAACL-04*, pages 273–280.

Dan Gildea. 2003. Loosely tree-based alignment for machine translation. In *Proceedings of ACL-03*.

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In *Proceedings of HLT/NAACL-04*, pages 105–112.

Mary Hearne and Andy Way. 2003. Seeing the wood for the trees: Data-Oriented Translation. In *Proceedings of MT Summit IX*.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL-03*, pages 423–430.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT/NAACL-03*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP-04*.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP-02*.

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

I. Dan Melamed. 2004. Statistical machine translation by parsing. In *Proceedings of ACL-04*, pages 653–660.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of ACL-00*.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL-02*.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: syntactically informed phrasal SMT. In *Proceedings of ACL-05*, pages 271–279.

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

Fei Xia and Michael McCord. 2004. Improving a statistical MT system with automatically learned rewrite patterns. In *Proceedings of COLING-04*.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of ACL-01*.

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 LREC-04*.

Grammatical Machine Translation

Stefan Riezler and **John T. Maxwell III**
Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto, CA 94304

Abstract

We present an approach to statistical machine translation that combines ideas from phrase-based SMT and traditional grammar-based MT. Our system incorporates the concept of multi-word translation units into transfer of dependency structure snippets, and models and trains statistical components according to state-of-the-art SMT systems. Compliant with classical transfer-based MT, target dependency structure snippets are input to a grammar-based generator. An experimental evaluation shows that the incorporation of a grammar-based generator into an SMT framework provides improved grammaticality while achieving state-of-the-art quality on in-coverage examples, suggesting a possible hybrid framework.

1 Introduction

Recent approaches to statistical machine translation (SMT) piggyback on the central concepts of phrase-based SMT (Och et al., 1999; Koehn et al., 2003) and at the same time attempt to improve some of its shortcomings by incorporating syntactic knowledge in the translation process. Phrase-based translation with multi-word units excels at modeling local ordering and short idiomatic expressions, however, it lacks a mechanism to learn long-distance dependencies and is unable to generalize to unseen phrases that share non-overt linguistic information. Publicly available statistical parsers can provide the syntactic information that is necessary for linguistic generalizations and for the resolution of non-local dependencies. This information source is deployed in recent work either for *pre-ordering* source sentences before they are input to to a phrase-based system (Xia and McCord, 2004; Collins et al., 2005), or for *re-ordering* the output of translation models by statistical ordering models that access linguistic information on dependencies and part-of-speech (Lin, 2004; Ding and Palmer, 2005; Quirk et al., 2005)[1].

While these approaches deploy dependency-style grammars for parsing source and/or target text, a utilization of grammar-based generation on the output of translation models has not yet been attempted in dependency-based SMT. Instead, simple target language realization models that can easily be trained to reflect the ordering of the reference translations in the training corpus are preferred. The advantage of such models over grammar-based generation seems to be supported, for example, by Quirk et al.'s (2005) improvements over phrase-based SMT as well as over an SMT system that deploys a grammar-based generator (Menezes and Richardson, 2001) on n-gram based automatic evaluation scores (Papineni et al., 2001; Doddington, 2002). Another data point, however, is given by Charniak et al. (2003) who show that parsing-based language modeling can improve grammaticality of translations, even if these improvements are not recorded under n-gram based evaluation measures.

[1] A notable exception to this kind of approach is Chiang (2005) who introduces syntactic information into phrase-based SMT via hierarchical phrases rather than by external parsing.

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 248–255,
New York, June 2006.

In this paper we would like to step away from n-gram based automatic evaluation scores for a moment, and investigate the possible contributions of incorporating a grammar-based generator into a dependency-based SMT system. We present a dependency-based SMT model that integrates the idea of multi-word translation units from phrase-based SMT into a transfer system for dependency structure snippets. The statistical components of our system are modeled on the phrase-based system of Koehn et al. (2003), and component weights are adjusted by minimum error rate training (Och, 2003). In contrast to phrase-based SMT and to the above cited dependency-based SMT approaches, our system feeds dependency-structure snippets into a grammar-based generator, and determines target language ordering by applying n-gram and distortion models after grammar-based generation. The goal of this ordering model is thus not foremost to reflect the ordering of the reference translations, but to improve the grammaticality of translations.

Since our system uses standard SMT techniques to learn about correct lexical choice and idiomatic expressions, it allows us to investigate the contribution of grammar-based generation to dependency-based SMT[2]. In an experimental evaluation on the test-set that was used in Koehn et al. (2003) we show that for examples that are in coverage of the grammar-based system, we can achieve state-of-the-art quality on n-gram based evaluation measures. To discern the factors of grammaticality and translational adequacy, we conducted a manual evaluation on 500 in-coverage and 500 out-of-coverage examples. This showed that an incorporation of a grammar-based generator into an SMT framework provides improved grammaticality over phrase-based SMT on in-coverage examples. Since in our system it is determinable whether an example is in-coverage, this opens the possibility for a hybrid system that achieves improved grammaticality at state-of-the-art translation quality.

[2]A comparison of the approaches of Quirk et al. (2005) and Menezes and Richardson (2001) with respect to ordering models is difficult because they differ from each other in their statistical and dependency-tree alignment models.

2 Extracting F-Structure Snippets

Our method for extracting transfer rules for dependency structure snippets operates on the paired sentences of a sentence-aligned bilingual corpus. Similar to phrase-based SMT, our approach starts with an improved word-alignment that is created by intersecting alignment matrices for both translation directions, and refining the intersection alignment by adding directly adjacent alignment points, and alignment points that align previously unaligned words (see Och et al. (1999)). Next, source and target sentences are parsed using source and target LFG grammars to produce a set of possible f(unctional) dependency structures for each side (see Riezler et al. (2002) for the English grammar and parser; Butt et al. (2002) for German). The two f-structures that most preserve dependencies are selected for further consideration. Selecting the most similar instead of the most probable f-structures is advantageous for rule induction since it provides for higher coverage with simpler rules. In the third step, the many-to-many word alignment created in the first step is used to define many-to-many correspondences between the substructures of the f-structures selected in the second step. The parsing process maintains an association between words in the string and particular predicate features in the f-structure, and thus the predicates on the two sides are implicitly linked by virtue of the original word alignment. The word alignment is extended to f-structures by setting into correspondence the f-structure units that immediately contain linked predicates. These f-structure correspondences are the basis for hypothesizing candidate transfer rules.

To illustrate, suppose our corpus contains the following aligned sentences (this example is taken from our experiments on German-to-English translation):

Dafür bin ich zutiefst dankbar.
I have a deep appreciation for that.

Suppose further that we have created the many-to-many bi-directional word alignment

Dafür{*6 7*} *bin*{*2*} *ich*{*1*} *zutiefst*{*3 4 5*}
dankbar{*5*}

indicating for example that *Dafür* is aligned with words 6 and 7 of the English sentence (*for* and *that*).

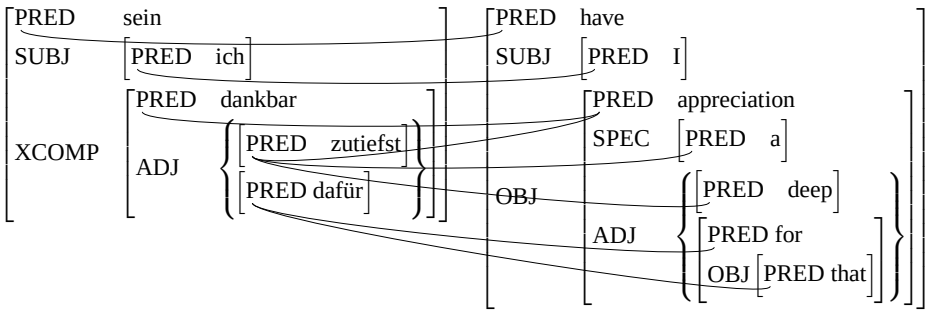


Figure 1: F-structure alignment for induction of German-to-English transfer rules.

This results in the links between the predicates of the source and target f-structures shown in Fig. 1.

From these source-target f-structure alignments transfer rules are extracted in two steps. In the first step, primitive transfer rules are extracted directly from the alignment of f-structure units. These include simple rules for mapping lexical predicates such as:

```
PRED(%X1, ich)  ==>  PRED(%X1, I)
```

and somewhat more complicated rules for mapping local f-structure configurations. For example, the rule shown below is derived from the alignment of the outermost f-structures. It maps any f-structure whose pred is *sein* to an f-structure with pred *have*, and in addition interprets the subj-to-subj link as an indication to map the subject of a source with this predicate into the subject of the target and the xcomp of the source into the object of the target. Features denoting number, person, type, etc. are not shown; variables %X denote f-structure values.

```
PRED(%X1,sein)          PRED(%X1,have)
SUBJ(%X1,%X2)    ==>    SUBJ(%X1,%X2)
XCOMP(%X1,%X3)          OBJ(%X1,%X3)
```

The following rule shows how a single source f-structure can be mapped to a local configuration of several units on the target side, in this case the single f-structure headed by *dafür* into one that corresponds to an English preposition+object f-structure.

```
                        PRED(%X1,for)
PRED(%X1, dafür)  ==>   OBJ(%X1,%X2)
                        PRED(%X2,that)
```

Transfer rules are required to only operate on contiguous units of the f-structure that are consistent with the word alignment. This *transfer contiguity constraint* states that

1. source and target f-structures are each connected.
2. f-structures in the transfer source can only be aligned with f-structures in the transfer target, and vice versa.

This constraint on f-structures is analogous to the constraint on contiguous and alignment-consistent phrases employed in phrase-based SMT. It prevents the extraction of a transfer rule that would translate *dankbar* directly into *appreciation* since *appreciation* is aligned also to *zutiefst* and its f-structure would also have to be included in the transfer. Thus, the primitive transfer rule for these predicates must be:

```
PRED(%X1,dankbar)          PRED(%X1,appr.)
ADJ(%X1,%X2)         ==>   SPEC(%X1,%X2)
in_set(%X3,%X2)            PRED(%X2,a)
PRED(%X3,zutiefst)         ADJ(%X1,%X3)
                           in_set(%X4,%X3)
                           PRED(%X4,deep)
```

In the second step, rules for more complex mappings are created by combining primitive transfer rules that are adjacent in the source and target f-structures. For instance, we can combine the primitive transfer rule that maps *sein* to *have* with the primitive transfer rule that maps *ich* to *I* to produce the complex transfer rule:

```
PRED(%X1,sein)          PRED(%X1,have)
SUBJ(%X1,%X2)    ==>    SUBJ(%X1,%X2)
PRED(%X2,ich)           PRED(%X2,I)
XCOMP(%X1,%X3)          OBJ(%X1,%X3)
```

In the worst case, there can be an exponential number of combinations of primitive transfer rules, so we only allow at most three primitive transfer rules to be combined. This produces $O(n^2)$ trans-

fer rules in the worst case, where n is the number of f-structures in the source.

Other points where linguistic information comes into play is in morphological stemming in f-structures, and in the optional filtering of f-structure phrases based on consistency of linguistic types. For example, the extraction of a phrase-pair that translates *zutiefst dankbar* into *a deep appreciation* is valid in the string-based world, but would be prevented in the f-structure world because of the incompatibility of the types *A* and *N* for adjectival *dankbar* and nominal *appreciation*. Similarly, a transfer rule translating *sein* to *have* could be dispreferred because of a mismatch in the the verbal types *V/A* and *V/N*. However, the transfer of *sein zutiefst dankbar* to *have a deep appreciation* is licensed by compatible head types *V*.

3 Parsing-Transfer-Generation

We use LFG grammars, producing c(onstituent)-structures (trees) and f(unctional)-structures (attribute value matrices) as output, for parsing source and target text (Riezler et al., 2002; Butt et al., 2002). To increase robustness, the standard grammar is augmented with a FRAGMENT grammar. This allows sentences that are outside the scope of the standard grammar to be parsed as well-formed chunks specified by the grammar, with unparsable tokens possibly interspersed. The correct parse is determined by a fewest-chunk method.

Transfer converts source into a target f-structures by non-deterministically applying all of the induced transfer rules in parallel. Each fact in the German f-structure must be transferred by exactly one transfer rule. For robustness a default rule is included that transfers any fact as itself. Similar to parsing, transfer works on a chart. The chart has an edge for each combination of facts that have been transferred. When the chart is complete, the outputs of the transfer rules are unified to make sure they are consistent (for instance, that the transfer rules did not produce two determiners for the same noun). Selection of the most probable transfer output is done by beam-decoding on the transfer chart.

LFG grammars can be used bidirectionally for parsing and generation, thus the existing English grammar used for parsing the training data can also be used for generation of English translations. For in-coverage examples, the grammar specifies c-structures that differ in linear precedence of subtrees for a given f-structure, and realizes the terminal yield according to morphological rules. In order to guarantee non-empty output for the overall translation system, the generation component has to be fault-tolerant in cases where the transfer system operates on a fragmentary parse, or produces non-valid f-structures from valid input f-structures. For generation from unknown predicates, a default morphology is used to inflect the source stem correctly for English. For generation from unknown structures, a default grammar is used that allows any attribute to be generated in any order as any category, with optimality marks set so as to prefer the standard grammar over the default grammar.

4 Statistical Models and Training

The statistical components of our system are modeled on the statistical components of the phrase-based system Pharaoh, described in Koehn et al. (2003) and Koehn (2004). Pharaoh integrates the following 8 statistical models: relative frequency of phrase translations in source-to-target and target-to-source direction, lexical weighting in source-to-target and target-to-source direction, phrase count, language model probability, word count, and distortion probability.

Correspondingly, our system computes the following statistics for each translation:

1. log-probability of source-to-target transfer rules, where the probability $r(\mathtt{e}|\mathtt{f})$ of a rule that transfers source snippet $\mathtt{f}$ into target snippet $\mathtt{e}$ is estimated by the relative frequency

$$r(\mathtt{e}|\mathtt{f}) = \frac{\text{count}(\mathtt{f} ==> \mathtt{e})}{\sum_{\mathtt{e}'} \text{count}(\mathtt{f} ==> \mathtt{e}')}$$

2. log-probability of target-to-source rules
3. log-probability of lexical translations from source to target snippets, estimated from Viterbi alignments $\hat{a}$ between source word positions $i = 1, \ldots, n$ and target word positions $j = 1, \ldots, m$ for stems f_i and e_j in snippets $\mathtt{f}$ and $\mathtt{e}$ with relative word translation frequen-

cies $t(e_j|f_i)$:

$$l(\mathtt{e}|\mathtt{f}) = \prod_j \frac{1}{|\{i|(i,j) \in \hat{a}\}|} \sum_{(i,j)\in\hat{a}} t(e_j|f_i)$$

4. log-probability of lexical translations from target to source snippets
5. number of transfer rules
6. number of transfer rules with frequency 1
7. number of default transfer rules (translating source features into themselves)
8. log-probability of strings of predicates from root to frontier of target f-structure, estimated from predicate trigrams in English f-structures
9. number of predicates in target f-structure
10. number of constituent movements during generation based on the original order of the head predicates of the constituents (for example, `AP[2] BP[3] CP[1]` counts as two movements since the head predicate of CP moved from the first position to the third position)
11. number of generation repairs
12. log-probability of target string as computed by trigram language model
13. number of words in target string

These statistics are combined into a log-linear model whose parameters are adjusted by minimum error rate training (Och, 2003).

5 Experimental Evaluation

The setup for our experimental comparison is German-to-English translation on the Europarl parallel data set[3]. For quick experimental turnaround we restricted our attention to sentences with 5 to 15 words, resulting in a training set of 163,141 sentences and a development set of 1967 sentences. Final results are reported on the test set of 1,755 sentences of length 5-15 that was used in Koehn et al. (2003). To extract transfer rules, an improved bidirectional word alignment was created for the training data from the word alignment of IBM model 4 as implemented by GIZA++ (Och et al., 1999). Training sentences were parsed using German and English LFG grammars (Riezler et al., 2002; Butt et al., 2002). The grammars obtain 100% coverage on unseen data. 80% are parsed as full parses; 20% receive FRAGMENT parses. Around 700,000 transfer rules were extracted from f-structures pairs chosen according to a dependency similarity measure. For language modeling, we used the trigram model of Stolcke (2002).

When applied to translating unseen text, the system operates on n-best lists of parses, transferred f-structures, and generated strings. For minimum-error-rate training on the development set, and for translating the test set, we considered 1 German parse for each source sentence, 10 transferred f-structures for each source parse, and 1,000 generated strings for each transferred f-structure. Selection of most probable translations proceeds in two steps: First, the most probable transferred f-structure is computed by a beam search on the transfer chart using the first 10 features described above. These features include tests on source and target f-structure snippets related via transfer rules (features 1-7) as well as language model and distortion features on the target c- and f-structures (features 8-10). In our experiments, the beam size was set to 20 hypotheses. The second step is based on features 11-13, which are computed on the strings that were actually generated from the selected n-best f-structures.

We compared our system to IBM model 4 as produced by GIZA++ (Och et al., 1999) and a phrase-based SMT model as provided by Pharaoh (2004). The same improved word alignment matrix and the same training data were used for phrase-extraction for phrase-based SMT as well as for transfer-rule extraction for LFG-based SMT. Minimum-error-rate training was done using Koehn's implementation of Och's (2003) minimum-error-rate model. To train the weights for phrase-based SMT we used the first 500 sentences of the development set; the weights of the LFG-based translator were adjusted on the 750 sentences that were in coverage of our grammars.

For automatic evaluation, we use the NIST metric (Doddington, 2002) combined with the approximate randomization test (Noreen, 1989), providing the desired combination of a sensitive evaluation metric and an accurate significance test (see Riezler and

[3] http://people.csail.mit.edu/koehn/publications/europarl/

Table 1: NIST scores on test set for IBM model 4 (M4), phrase-based SMT (P), and the LFG-based SMT (LFG) on the full test set and on in-coverage examples for LFG. Results in the same row that are not statistically significant from each other are marked with a $*$.

	M4	LFG	P
in-coverage	5.13	*5.82	*5.99
full test set	*5.57	*5.62	6.40

Table 2: Preference ratings of two human judges for translations of phrase-based SMT (P) or LFG-based SMT (LFG) under criteria of fluency/grammaticality and translational/semantic adequacy on 500 in-coverage examples. Ratings by judge 1 are shown in rows, for judge 2 in columns. Agreed-on examples are shown in boldface in the diagonals.

	adequacy			grammaticality		
j1\j2	P	LFG	equal	P	LFG	equal
P	**48**	8	7	**36**	2	9
LFG	10	**105**	18	6	**113**	17
equal	53	60	**192**	51	44	**223**

Maxwell (2005)). In order to avoid a random assessment of statistical significance in our three-fold pairwise comparison, we reduce the per-comparison significance level to 0.01 so as to achieve a standard experimentwise significance level of 0.05 (see Cohen (1995)). Table 1 shows results for IBM model 4, phrase-based SMT, and LFG-based SMT, where examples that are in coverage of the LFG-based systems are evaluated separately. Out of the 1,755 sentences of the test set, 44% were in coverage of the LFG-grammars; for 51% the system had to resort to the FRAGMENT technique for parsing and/or repair techniques in generation; in 5% of the cases our system timed out. Since our grammars are not set up with punctuation in mind, punctuation is ignored in all evaluations reported below.

For in-coverage examples, the difference between NIST scores for the LFG system and the phrase-based system is statistically not significant. On the full set of test examples, the suboptimal quality on out-of-coverage examples overwhelms the quality achieved on in-coverage examples, resulting in a statistically not significant result difference in NIST scores between the LFG system and IBM model 4.

In order to discern the factors of grammaticality and translational adequacy, we conducted a manual evaluation on randomly selected 500 examples that were in coverage of the grammar-based generator. Two independent human judges were presented with the source sentence, and the output of the phrase-based and LFG-based systems in a blind test. This was achieved by displaying the system outputs in random order. The judges were asked to indicate a preference for one system translation over the other, or whether they thought them to be of equal quality. These questions had to be answered separately under the criteria of grammaticality/fluency and translational/semantic adequacy. As shown in Table 2, both judges express a preference for the LFG system over the phrase-based system for both adequacy and grammaticality. If we just look at sentences where judges agree, we see a net improvement on translational adequacy of 57 sentences, which is an improvement of 11.4% over the 500 sentences. If this were part of a hybrid system, this would amount to a 5% overall improvement in translational adequacy. Similarly we see a net improvement on grammaticality of 77 sentences, which is an improvement of 15.4% over the 500 sentences or 6.7% overall in a hybrid system. Result differences on agreed-on ratings are statistically significant, where significance was assessed by approximate randomization via stratified shuffling of the preferences between the systems (Noreen, 1989). Examples from the manual evaluation are shown in Fig. 2.

Along the same lines, a further manual evaluation was conducted on 500 randomly selected examples that were out of coverage of the LFG-based grammars. Across the combined set of 1,000 in-coverage and out-of-coverage sentences, this resulted in an agreed-on preference for the phrase-based system in 204 cases and for the LFG-based system in 158 cases under the measure of translational adequacy. Under the grammaticality measure the phrase-based system was preferred by both judges in 157 cases and the LFG-based system in 136 cases.

6 Discussion

The above presented evaluation of the LFG-based translator shows promising results for examples that are in coverage of the employed LFG grammars. However, a back-off to robustness techniques in parsing and/or generation results in a considerable

(1)	src:	in diesem fall werde ich meine verantwortung wahrnehmen
	ref:	then i will exercise my responsibility
	LFG:	in this case i accept my responsibility
	P:	in this case i shall my responsibilities
(2)	src:	die politische stabilität hängt ab von der besserung der lebensbedingungen
	ref:	political stability depends upon the improvement of living conditions
	LFG:	the political stability hinges on the recovery the conditions
	P:	the political stability is rejects the recovery of the living conditions
(3)	src:	und schließlich muß dieser agentur eine kritische haltung gegenüber der kommission selbst erlaubt sein
	ref:	moreover the agency must be able to criticise the commission itself
	LFG:	and even to the commission a critical stance must finally be allowed this agency
	P:	finally this is a critical attitude towards the commission itself to be agency
(4)	src:	nach der ratifizierung werden co2 emissionen ihren preis haben
	ref:	after ratification co2 emission will have a price tag
	LFG:	carbon dioxide emissions have its price following the ratification
	P:	after the ratification co2 emissions are a price
(5)	src:	die lebensmittel müssen die sichere ernährung des menschen gewährleisten
	ref:	man's food must be safe to eat
	LFG:	food must guarantee the safe nutrition of the people
	P:	the people of the nutrition safe food must guarantee
(6)	src:	was wir morgen beschließen werden ist letztlich material für das vermittlungsverfahren
	ref:	whatever we agree tomorrow will ultimately have to go into the conciliation procedure
	LFG:	one tomorrow we approved what is ultimately material for the conciliation procedure
	P:	what we decide tomorrow is ultimately material for the conciliation procedure
(7)	src:	die verwaltung muß künftig schneller reagieren können
	ref:	in future the administration must be able to react more quickly
	LFG:	more in future the administration must be able to react
	P:	the administration must be able to react more quickly
(8)	src:	das ist jetzt über 40 jahre her
	ref:	that was over 40 years ago
	LFG:	on 40 years ago it is now
	P:	that is now over 40 years ago
(9)	src:	das ist schon eine seltsame vorstellung von gleichheit
	ref:	a strange notion of equality
	LFG:	equality that is even a strange idea
	P:	this is already a strange idea of equality
(10)	src:	frau präsidentin ich beglückwünsche herrn nicholson zu seinem ausgezeichneten bericht
	ref:	madam president i congratulate mr nicholson on his excellent report
	LFG:	madam president i congratulate mister nicholson on his report excellented
	P:	madam president i congratulate mr nicholson for his excellent report

Figure 2: Examples from manual evaluation: Preference for LFG-based system (LFG) over phrase-based system (P) under both adequacy and grammaticality (ex 1-5), preference of phrased-based system over LFG (6-10) , together with source (src) sentences and human reference (ref) translations. All ratings are agreed on by both judges.

loss in translation quality. The high percentage of examples that fall out of coverage of the LFG-based system can partially be explained by the accumulation of errors in parsing the training data where source and target language parser each produce FRAGMENT parses in 20% of the cases. Together with errors in rule extraction, this results in a large number ill-formed transfer rules that force the generator to back-off to robustness techniques. In applying the parse-transfer-generation pipeline to translating unseen text, parsing errors can cause erroneous transfer, which can result in generation errors. Similar effects can be observed for errors in translating in-coverage examples. Here disambiguation errors in parsing and transfer propagate through the system, producing suboptimal translations. An error analysis on 100 suboptimal in-coverage examples from the development set showed that 69 suboptimal translations were due to transfer errors, 10 of which were due to errors in parsing.

The discrepancy between NIST scores and manual preference rankings can be explained on the one hand by the suboptimal integration of transfer and generation in our system, making it infeasible to work with large n-best lists in training and application. Moreover, despite our use of minimum-error-

rate training and n-gram language models, our system cannot be adjusted to maximize n-gram scores on reference translation in the same way as phrase-based systems since statistical ordering models are employed in our framework *after* grammar-based generation, thus giving preference to grammaticality over similarity to reference translations.

7 Conclusion

We presented an SMT model that marries phrase-based SMT with traditional grammar-based MT by incorporating a grammar-based generator into a dependency-based SMT system. Under the NIST measure, we achieve results in the range of the state-of-the-art phrase-based system of Koehn et al. (2003) for in-coverage examples of the LFG-based system. A manual evaluation of a large set of such examples shows that on in-coverage examples our system achieves significant improvements in grammaticality and also translational adequacy over the phrase-based system. Fortunately, it is determinable when our system is in-coverage, which opens the possibility for a hybrid system that achieves improved grammaticality at state-of-the-art translation quality. Future work thus will concentrate on improvements of in-coverage translations e.g., by stochastic generation. Furthermore, we intend to apply our system to other language pairs and larger data sets.

Acknowledgements

We would like to thank Sabine Blum for her invaluable help with the manual evaluation.

References

Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The parallel grammar project. *COLING'02, Workshop on Grammar Engineering and Evaluation.*

Eugene Charniak, Kevin Knight, and Kenji Yamada. 2003. Syntax-based language models for statistical machine translation. *MT Summit IX*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. *ACL'05*.

Paul R. Cohen. 1995. *Empirical Methods for Artificial Intelligence*. The MIT Press.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. *ACL'05*.

Yuan Ding and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. *ACL'05*.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. *ARPA Workshop on Human Language Technology*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. *HLT-NAACL'03*.

Philipp Koehn. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. User manual. Technical report, USC ISI.

Dekang Lin. 2004. A path-based transfer model for statistical machine translation. *COLING'04*.

Arul Menezes and Stephen D. Richardson. 2001. A best-first alignment algorithm for automatic extraction of transfer-mappings from bilingual corpora. *Workshop on Data-Driven Machine Translation.*

Eric W. Noreen. 1989. *Computer Intensive Methods for Testing Hypotheses. An Introduction*. Wiley.

Franz Josef Och, Christoph Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. *EMNLP'99*.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. *HLT-NAACL'03*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. Technical Report IBM RC22176 (W0190-022).

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. *ACL'05*.

Stefan Riezler and John Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for mt. *ACL-05 Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization.*

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. *ACL'02*.

Stefan Riezler, Tracy H. King, Richard Crouch, and Annie Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. *HLT-NAACL'03*.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. *International Conference on Spoken Language Processing*.

Fei Xia and Michael McCord. 2004. Improving a statistical mt system with automatically learned rewrite patterns. *COLING'04*.

Synchronous Binarization for Machine Translation

Hao Zhang
Computer Science Department
University of Rochester
Rochester, NY 14627
zhanghao@cs.rochester.edu

Liang Huang
Dept. of Computer & Information Science
University of Pennsylvania
Philadelphia, PA 19104
lhuang3@cis.upenn.edu

Daniel Gildea
Computer Science Department
University of Rochester
Rochester, NY 14627
gildea@cs.rochester.edu

Kevin Knight
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
knight@isi.edu

Abstract

Systems based on synchronous grammars and tree transducers promise to improve the quality of statistical machine translation output, but are often very computationally intensive. The complexity is exponential in the size of individual grammar rules due to arbitrary re-orderings between the two languages, and rules extracted from parallel corpora can be quite large. We devise a linear-time algorithm for factoring syntactic re-orderings by binarizing synchronous rules when possible and show that the resulting rule set significantly improves the speed and accuracy of a state-of-the-art syntax-based machine translation system.

1 Introduction

Several recent syntax-based models for machine translation (Chiang, 2005; Galley et al., 2004) can be seen as instances of the general framework of synchronous grammars and tree transducers. In this framework, both alignment (*synchronous parsing*) and decoding can be thought of as parsing problems, whose complexity is in general exponential in the number of nonterminals on the right hand side of a grammar rule. To alleviate this problem, we investigate bilingual binarization to factor the synchronous grammar to a smaller branching factor, although it is not guaranteed to be successful for any synchronous rule with arbitrary permutation. In particular:

- We develop a technique called *synchronous binarization* and devise a fast binarization algorithm such that the resulting rule set allows efficient algorithms for both synchronous parsing and decoding with integrated n-gram language models.
- We examine the effect of this binarization method on end-to-end machine translation quality, compared to a more typical baseline method.
- We examine cases of non-binarizable rules in a large, empirically-derived rule set, and we investigate the effect on translation quality when excluding such rules.

Melamed (2003) discusses binarization of multitext grammars on a theoretical level, showing the importance and difficulty of binarization for efficient synchronous parsing. One way around this difficulty is to stipulate that all rules must be binary from the outset, as in inversion-transduction grammar (ITG) (Wu, 1997) and the binary synchronous context-free grammar (SCFG) employed by the Hiero system (Chiang, 2005) to model the hierarchical phrases. In contrast, the rule extraction method of Galley et al. (2004) aims to incorporate more syntactic information by providing parse trees for the target language and extracting tree transducer rules that apply to the parses. This approach results in rules with many nonterminals, making good binarization techniques critical.

Suppose we have the following SCFG, where superscripts indicate reorderings (formal definitions of

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 256–263,
New York, June 2006.

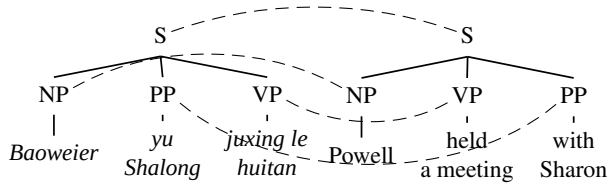


Figure 1: A pair of synchronous parse trees in the SCFG (1). The dashed curves indicate pairs of synchronous nonterminals (and sub trees).

SCFGs can be found in Section 2):

$$
(1)\quad
\begin{array}{rl}
\text{S} \rightarrow & \text{NP}^{(1)}\ \text{VP}^{(2)}\ \text{PP}^{(3)},\ \text{NP}^{(1)}\ \text{PP}^{(3)}\ \text{VP}^{(2)} \\
\text{NP} \rightarrow & \text{Powell},\ \textit{Baoweier} \\
\text{VP} \rightarrow & \text{held a meeting},\ \textit{juxing le huitan} \\
\text{PP} \rightarrow & \text{with Sharon},\ \textit{yu Shalong}
\end{array}
$$

Decoding can be cast as a (monolingual) parsing problem since we only need to parse the source-language side of the SCFG, as if we were constructing a CFG projected on Chinese out of the SCFG. The only extra work we need to do for decoding is to build corresponding target-language (English) subtrees in parallel. In other words, we build *synchronous trees* when parsing the source-language input, as shown in Figure 1.

To efficiently decode with CKY, we need to binarize the projected CFG grammar.[1] Rules can be binarized in different ways. For example, we could binarize the first rule left to right or right to left:

$$
\begin{array}{rl}
\text{S} \rightarrow & V_{\text{NP-PP}}\ \text{VP} \\
V_{\text{NP-PP}} \rightarrow & \text{NP PP}
\end{array}
\quad \text{or} \quad
\begin{array}{rl}
\text{S} \rightarrow & \text{NP}\ V_{\text{PP-VP}} \\
V_{\text{PP-VP}} \rightarrow & \text{PP VP}
\end{array}
$$

We call those intermediate symbols (e.g. $V_{\text{PP-VP}}$) *virtual nonterminals* and corresponding rules *virtual rules*, whose probabilities are all set to 1.

These two binarizations are no different in the translation-model-only decoding described above, just as in monolingual parsing. However, in the source-channel approach to machine translation, we need to combine probabilities from the translation model (an SCFG) with the language model (an n-gram), which has been shown to be very important for translation quality (Chiang, 2005). To do bigram-integrated decoding, we need to augment each chart item (X, i, j) with two target-language *boundary words* u and v to produce a bigram-item like $\left(\begin{smallmatrix} u \cdots v \\ i \quad\ j \end{smallmatrix}\right)$, following the dynamic programming algorithm of Wu (1996).

Now the two binarizations have very different effects. In the first case, we first combine NP with PP:

$$
\frac{\left(\begin{smallmatrix} \text{Powell} & \cdots & \text{Powell} \\ 1 & \text{NP} & 2 \end{smallmatrix}\right) : p \qquad \left(\begin{smallmatrix} \text{with} & \cdots & \text{Sharon} \\ 2 & \text{PP} & 4 \end{smallmatrix}\right) : q}{\left(\begin{smallmatrix} \text{Powell} \cdots \text{Powell} & \cdots & \text{with} \cdots \text{Sharon} \\ 1 & V_{\text{NP-PP}} & 4 \end{smallmatrix}\right) : pq}
$$

where p and q are the scores of antecedent items.

This situation is unpleasant because in the target-language NP and PP are *not* contiguous so we cannot apply language model scoring when we build the $V_{\text{NP-PP}}$ item. Instead, we have to maintain all four boundary words (rather than two) and postpone the language model scoring till the next step where $V_{\text{NP-PP}}$ is combined with $\left(\begin{smallmatrix} \text{held} & \cdots & \text{meeting} \\ 2 & \text{VP} & 4 \end{smallmatrix}\right)$ to form an S item. We call this binarization method *monolingual binarization* since it works only on the source-language projection of the rule without respecting the constraints from the other side.

This scheme generalizes to the case where we have n nonterminals in a SCFG rule, and the decoder conservatively assumes nothing can be done on language model scoring (because target-language spans are non-contiguous in general) until the real nonterminal has been recognized. In other words, target-language boundary words from each child nonterminal of the rule will be cached in all virtual nonterminals derived from this rule. In the case of m-gram integrated decoding, we have to maintain $2(m-1)$ boundary words for each child nonterminal, which leads to a prohibitive overall complexity of $O(|w|^{3+2n(m-1)})$, which is exponential in rule size (Huang et al., 2005). Aggressive pruning must be used to make it tractable in practice, which in general introduces many search errors and adversely affects translation quality.

In the second case, however:

$$
\frac{\left(\begin{smallmatrix} \text{with} & \cdots & \text{Sharon} \\ 2 & \text{PP} & 4 \end{smallmatrix}\right) : r \qquad \left(\begin{smallmatrix} \text{held} & \cdots & \text{meeting} \\ 4 & \text{VP} & 7 \end{smallmatrix}\right) : s}{\left(\begin{smallmatrix} \text{held} & \cdots & \text{Sharon} \\ 2 & V_{\text{PP-VP}} & 7 \end{smallmatrix}\right) : rs \cdot \Pr(\text{with} \mid \text{meeting})}
$$

Here since PP and VP are contiguous (but swapped) in the target-language, we can include the

[1] Other parsing strategies like the Earley algorithm use an internal binary representation (e.g. dotted-rules) of the original grammar to ensure cubic time complexity.

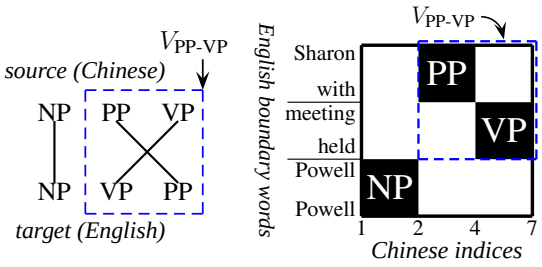


Figure 2: The alignment pattern (left) and alignment matrix (right) of the synchronous production.

language model score by adding $\Pr(\text{with} \mid \text{meeting})$, and the resulting item again has two boundary words. Later we add $\Pr(\text{held} \mid \text{Powell})$ when the resulting item is combined with $\binom{\text{Powell}\ \cdots\ \text{Powell}}{1\qquad\quad 2}$ to form an S item. As illustrated in Figure 2, $V_{\text{PP-VP}}$ has contiguous spans on both source and target sides, so that we can generate a binary-branching SCFG:

$$\begin{aligned} \text{S} &\rightarrow \text{NP}^{(1)}\ V_{\text{PP-VP}}{}^{(2)},\ \text{NP}^{(1)}\ V_{\text{PP-VP}}{}^{(2)} \\ V_{\text{PP-VP}} &\rightarrow \text{VP}^{(1)}\ \text{PP}^{(2)},\ \text{PP}^{(2)}\ \text{VP}^{(1)} \end{aligned} \tag{2}$$

In this case m-gram integrated decoding can be done in $O(|w|^{3+4(m-1)})$ time which is much lower-order polynomial and no longer depends on rule size (Wu, 1996), allowing the search to be much faster and more accurate facing pruning, as is evidenced in the Hiero system of Chiang (2005) where he restricts the hierarchical phrases to be a binary SCFG. The benefit of binary grammars also lies in synchronous parsing (alignment). Wu (1997) shows that parsing a binary SCFG is in $O(|w|^6)$ while parsing SCFG is NP-hard in general (Satta and Peserico, 2005).

The same reasoning applies to tree transducer rules. Suppose we have the following tree-to-string rules, following Galley et al. (2004):

$$\begin{aligned} &\text{S}(x_0\text{:NP}, \text{VP}(x_2\text{:VP}, x_1\text{:PP})) \rightarrow x_0\ x_1\ x_2 \\ &\text{NP(NNP(Powell))} \rightarrow \textit{Baoweier} \\ &\text{VP(VBD(held), NP(DT(a) NPS(meeting)))} \\ &\qquad\qquad \rightarrow \textit{juxing le huitan} \\ &\text{PP(TO(with), NP(NNP(Sharon)))} \rightarrow \textit{yu Shalong} \end{aligned} \tag{3}$$

where the reorderings of nonterminals are denoted by variables x_i.

Notice that the first rule has a multi-level left-hand side subtree. This system can model non-isomorphic transformations on English parse trees to "fit" another language, for example, learning that the $(S\ (V\ O))$ structure in English should be transformed into a $(V\ S\ O)$ structure in Arabic, by looking at two-level tree fragments (Knight and Graehl, 2005). From a synchronous rewriting point of view, this is more akin to synchronous tree substitution grammar (STSG) (Eisner, 2003). This larger locality is linguistically motivated and leads to a better parameter estimation. By imagining the left-hand-side trees as special nonterminals, we can virtually create an SCFG with the same generative capacity. The technical details will be explained in Section 3.2.

In general, if we are given an arbitrary synchronous rule with many nonterminals, what are the good decompositions that lead to a binary grammar? Figure 2 suggests that a binarization is good if every virtual nonterminal has contiguous spans on both sides. We formalize this idea in the next section.

2 Synchronous Binarization

A **synchronous CFG** (SCFG) is a context-free rewriting system for generating string pairs. Each rule (*synchronous production*) rewrites a nonterminal in two dimensions subject to the constraint that the sequence of nonterminal children on one side is a permutation of the nonterminal sequence on the other side. Each co-indexed child nonterminal pair will be further rewritten as a unit.[2] We define the language $L(G)$ produced by an SCFG G as the pairs of terminal strings produced by rewriting exhaustively from the start symbol.

As shown in Section 3.2, terminals do not play an important role in binarization. So we now write rules in the following notation:

$$X \rightarrow X_1^{(1)} ... X_n^{(n)},\ X_{\pi(1)}^{(\pi(1))} ... X_{\pi(n)}^{(\pi(n))}$$

where each X_i is a variable which ranges over nonterminals in the grammar and π is the **permutation** of the rule. We also define an SCFG rule as n-ary if its permutation is of n and call an SCFG n-ary if its longest rule is n-ary. Our goal is to produce an equivalent *binary* SCFG for an input n-ary SCFG.

[2] In making one nonterminal play dual roles, we follow the definitions in (Aho and Ullman, 1972; Chiang, 2005), originally known as Syntax Directed Translation Schema (SDTS). An alternative definition by Satta and Peserico (2005) allows co-indexed nonterminals taking different symbols in two dimensions. Formally speaking, we can construct an equivalent SDTS by creating a cross-product of nonterminals from two sides. See (Satta and Peserico, 2005, Sec. 4) for other details.

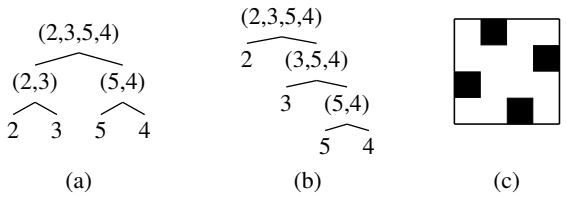


Figure 3: (a) and (b): two binarization patterns for $(2, 3, 5, 4)$. (c): alignment matrix for the non-binarizable permuted sequence $(2, 4, 1, 3)$

However, not every SCFG can be binarized. In fact, the binarizability of an n-ary rule is determined by the structure of its permutation, which can sometimes be resistant to factorization (Aho and Ullman, 1972). So we now start to rigorously define the binarizability of permutations.

2.1 Binarizable Permutations

A **permuted sequence** is a permutation of consecutive integers. For example, $(3, 5, 4)$ is a permuted sequence while $(2, 5)$ is not. As special cases, single numbers are permuted sequences as well.

A sequence $\mathbf{a}$ is said to be **binarizable** if it is a permuted sequence and either

1. $\mathbf{a}$ is a singleton, i.e. $\mathbf{a} = (a)$, or
2. $\mathbf{a}$ can be split into two sub sequences, i.e. $\mathbf{a} = (\mathbf{b}; \mathbf{c})$, where $\mathbf{b}$ and $\mathbf{c}$ are both binarizable permuted sequences. We call such a division $(\mathbf{b}; \mathbf{c})$ a **binarizable split** of $\mathbf{a}$.

This is a recursive definition. Each binarizable permuted sequence has at least one hierarchical binarization pattern. For instance, the permuted sequence $(2, 3, 5, 4)$ is binarizable (with two possible binarization patterns) while $(2, 4, 1, 3)$ is not (see Figure 3).

2.2 Binarizable SCFG

An SCFG is said to be **binarizable** if the permutation of each synchronous production is binarizable. We denote the class of binarizable SCFGs as **bSCFG**. This set represents an important subclass of SCFG that is easy to handle (parsable in $O(|w|^6)$) and covers many interesting longer-than-two rules.[3]

[3]Although we factor the SCFG rules individually and define bSCFG accordingly, there are some grammars (the dashed circle in Figure 4), which can be binarized only by analyzing interactions between rules. Below is a simple example:

S → X(1) X(2) X(3) X(4), X(2) X(4) X(1) X(3)
X → a , a

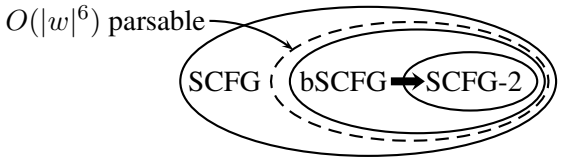


Figure 4: Subclasses of SCFG. The thick arrow denotes the direction of synchronous binarization. For clarity reasons, binary SCFG is coded as SCFG-2.

Theorem 1. *For each grammar G in bSCFG, there exists a binary SCFG G', such that $L(G') = L(G)$.*

Proof. Once we decompose the permutation of n in the original rule into binary permutations, all that remains is to decorate the skeleton binary parse with nonterminal symbols and attach terminals to the skeleton appropriately. We explain the technical details in the next section. □

3 Binarization Algorithms

We have reduced the problem of binarizing an SCFG rule into the problem of binarizing its permutation. This problem can be cast as an instance of synchronous ITG parsing (Wu, 1997). Here the parallel string pair that we are parsing is the integer sequence $(1...n)$ and its permutation $(\pi(1)...\pi(n))$. The goal of the ITG parsing is to find a synchronous tree that agrees with the alignment indicated by the permutation. In fact, as demonstrated previously, some permutations may have more than one binarization patterns among which we only need one. Wu (1997, Sec. 7) introduces a non-ambiguous ITG that prefers left-heavy binary trees so that for each permutation there is a unique synchronous derivation (binarization pattern).

However, this problem has more efficient solutions. Shapiro and Stephens (1991, p. 277) informally present an iterative procedure where in each pass it scans the permuted sequence from left to right and combines two adjacent sub sequences whenever possible. This procedure produces a left-heavy binarization tree consistent with the unambiguous ITG and runs in $O(n^2)$ time since we need n passes in the worst case. We modify this procedure and improve

iteration	stack	input	action	
		1 5 3 4 2		
	1	5 3 4 2	shift	
1	1 5	3 4 2	shift	
2	1 5 3	4 2	shift	
3	1 5 3 4	2	shift	
	1 5 3-4	2	reduce	$[3,4]$
	1 5-5	2	reduce	$\langle 5,[3,4]\rangle$
4	1 3-5 2		shift	
	1 2-5		reduce	$\langle 2,\langle 5,[3,4]\rangle\rangle$
	1-5		reduce	$[1,\langle 2,\langle 5,[3,4]\rangle\rangle]$

Figure 5: Example of Algorithm 1 on the input $(1,5,3,4,2)$. The rightmost column shows the binarization-trees generated at each reduction step.

it into a linear-time shift-reduce algorithm that only needs one pass through the sequence.

3.1 The linear-time skeleton algorithm

The (unique) **binarization tree** $bi(\mathbf{a})$ for a binarizable permuted sequence $\mathbf{a}$ is recursively defined as follows:

- if $\mathbf{a} = (a)$, then $bi(\mathbf{a}) = a$;
- otherwise let $\mathbf{a} = (\mathbf{b}; \mathbf{c})$ to be the *rightmost* binarizable split of $\mathbf{a}$. then

$$bi(\mathbf{a}) = \begin{cases} [bi(\mathbf{b}), bi(\mathbf{c})] & b_1 < c_1 \\ \langle bi(\mathbf{b}), bi(\mathbf{c})\rangle & b_1 > c_1. \end{cases}$$

For example, the binarization tree for $(2,3,5,4)$ is $[[2,3],\langle 5,4\rangle]$, which corresponds to the binarization pattern in Figure 3(a). We use [] and $\langle\rangle$ for straight and inverted combinations respectively, following the ITG notation (Wu, 1997). The rightmost split ensures left-heavy binary trees.

The skeleton binarization algorithm is an instance of the widely used left-to-right shift-reduce algorithm. It maintains a stack for contiguous subsequences discovered so far, like 2-5, 1. In each iteration, it shifts the next number from the input and repeatedly tries to reduce the top two elements on the stack if they are consecutive. See Algorithm 1 for details and Figure 5 for an example.

Theorem 2. *Algorithm 1 succeeds if and only if the input permuted sequence* $\mathbf{a}$ *is binarizable, and in case of success, the binarization pattern recovered is the binarization tree of* $\mathbf{a}$.

Proof. $\rightarrow$: it is obvious that if the algorithm succeeds then $\mathbf{a}$ is binarizable using the binarization pattern recovered.

$\leftarrow$: by a complete induction on n, the length of $\mathbf{a}$.

Base case: $n = 1$, trivial.

Assume it holds for all $n' < n$.

If $\mathbf{a}$ is binarizable, then let $\mathbf{a} = (\mathbf{b}; \mathbf{c})$ be its rightmost binarizable split. By the induction hypothesis, the algorithm succeeds on the partial input $\mathbf{b}$, reducing it to the single element $s[0]$ on the stack and recovering its binarization tree $bi(\mathbf{b})$.

Let $\mathbf{c} = (\mathbf{c}_1; \mathbf{c}_2)$. If $\mathbf{c}_1$ is binarizable and triggers our binarizer to make a straight combination of $(\mathbf{b}; \mathbf{c}_1)$, based on the property of permutations, it must be true that $(\mathbf{c}_1; \mathbf{c}_2)$ is a valid straight concatenation. We claim that $\mathbf{c}_2$ must be binarizable in this situation. So, $(\mathbf{b}, \mathbf{c}_1; \mathbf{c}_2)$ is a binarizable split to the right of the rightmost binarizable split $(\mathbf{b}; \mathbf{c})$, which is a contradiction. A similar contradiction will arise if $\mathbf{b}$ and $\mathbf{c}_1$ can make an inverted concatenation.

Therefore, the algorithm will scan through the whole $\mathbf{c}$ *as if* from the empty stack. By the induction hypothesis again, it will reduce $\mathbf{c}$ into $s[1]$ on the stack and recover its binarization tree $bi(\mathbf{c})$. Since $\mathbf{b}$ and $\mathbf{c}$ are combinable, the algorithm reduces $s[0]$ and $s[1]$ in the last step, forming the binarization tree for $\mathbf{a}$, which is either $[bi(\mathbf{b}), bi(\mathbf{c})]$ or $\langle bi(\mathbf{b}), bi(\mathbf{c})\rangle$. □

The running time of Algorithm 1 is linear in n, the length of the input sequence. This is because there are exactly n shifts and at most $n-1$ reductions, and each shift or reduction takes $O(1)$ time.

3.2 Binarizing tree-to-string transducers

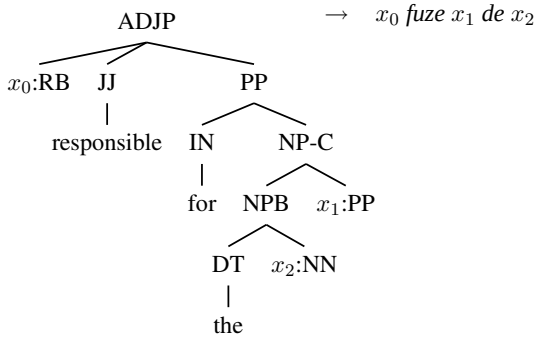
Without loss of generality, we have discussed how to binarize synchronous productions involving only nonterminals through binarizing the corresponding skeleton permutations. We still need to tackle a few technical problems in the actual system.

First, we are dealing with tree-to-string transducer rules. We view each left-hand side subtree as a monolithic nonterminal symbol and factor each transducer rule into two SCFG rules: one from the root nonterminal to the subtree, and the other from the subtree to the leaves. In this way we can uniquely reconstruct the tree-to-string derivation using the two-step SCFG derivation. For example,

Algorithm 1 The Linear-time Binarization Algorithm

function BINARIZABLE(**a**)
  $top \leftarrow 0$ ▷ stack top pointer
  PUSH(a_1, a_1) ▷ initial shift
  for $i \leftarrow 2$ **to** $|\mathbf{a}|$ **do** ▷ for each remaining element
    PUSH(a_i, a_i) ▷ shift
    while $top > 1$ **and** CONSECUTIVE$(s[top], s[top-1])$ **do** ▷ keep reducing if possible
      $(p, q) \leftarrow$ COMBINE$(s[top], s[top-1])$
      $top \leftarrow top - 2$
      PUSH(p, q)
  return $(top = 1)$ ▷ if reduced to a single element then the input is binarizable, otherwise not
function CONSECUTIVE$((a, b), (c, d))$
  return $(b = c - 1)$ **or** $(d = a - 1)$ ▷ either straight or inverted
function COMBINE$((a, b), (c, d))$
  return $(min(a, c), max(b, d))$

consider the following tree-to-string rule:



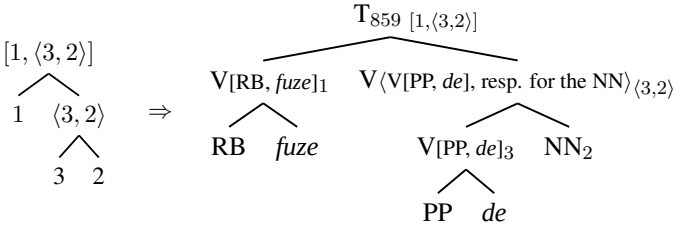
We create a specific nonterminal, say, T_{859}, which is a unique identifier for the left-hand side subtree and generate the following two SCFG rules:

$$\text{ADJP} \rightarrow \mathrm{T}_{859}\ ^{(1)}, \mathrm{T}_{859}\ ^{(1)}$$

$$\mathrm{T}_{859} \rightarrow \begin{array}{l}\mathrm{RB}^{(1)} \text{ resp. for the } \mathrm{NN}^{(2)}\ \mathrm{PP}^{(3)}, \\ \mathrm{RB}^{(1)}\ \textit{fuze}\ \mathrm{PP}^{(3)}\ \textit{de}\ \mathrm{NN}^{(2)}\end{array}$$

Second, besides synchronous nonterminals, terminals in the two languages can also be present, as in the above example. It turns out we can attach the terminals to the skeleton parse for the synchronous nonterminal strings quite freely as long as we can uniquely reconstruct the original rule from its binary parse tree. In order to do so we need to keep track of sub-alignments including both aligned nonterminals and neighboring terminals.

When binarizing the second rule above, we first run the skeleton algorithm to binarize the underlying permutation $(1, 3, 2)$ to its binarization tree $[1, \langle 3, 2\rangle]$. Then we do a post-order traversal to the skeleton tree, combining Chinese terminals (one at a time) at the leaf nodes and merging English terminals greedily at internal nodes:



A pre-order traversal of the decorated binarization tree gives us the following binary SCFG rules:

$$\begin{array}{lcl}\mathrm{T}_{859} & \rightarrow & \mathrm{V}_1{}^{(1)}\ \mathrm{V}_2{}^{(2)},\ \mathrm{V}_1{}^{(1)}\ \mathrm{V}_2{}^{(2)} \\ \mathrm{V}_1 & \rightarrow & \mathrm{RB}^{(1)},\ \mathrm{RB}^{(1)}\ \textit{fuze} \\ \mathrm{V}_2 & \rightarrow & \text{resp. for the } \mathrm{NN}^{(1)}\ \mathrm{V}_3^{(2)},\ \mathrm{V}_3^{(2)}\ \mathrm{NN}^{(1)} \\ \mathrm{V}_3 & \rightarrow & \mathrm{PP}^{(1)},\ \mathrm{PP}^{(1)}\ \textit{de}\end{array}$$

where the virtual nonterminals are:

V_1: V[RB, *fuze*]
V_2: V$\langle$V[PP, *de*], resp. for the NN$\rangle$
V_3: V[PP, *de*]

Analogous to the "dotted rules" in Earley parsing for monolingual CFGs, the names we create for the virtual nonterminals reflect the underlying sub-alignments, ensuring intermediate states can be shared across different tree-to-string rules without causing ambiguity.

The whole binarization algorithm still runs in time linear in the number of symbols in the rule (including both terminals and nonterminals).

4 Experiments

In this section, we answer two empirical questions.

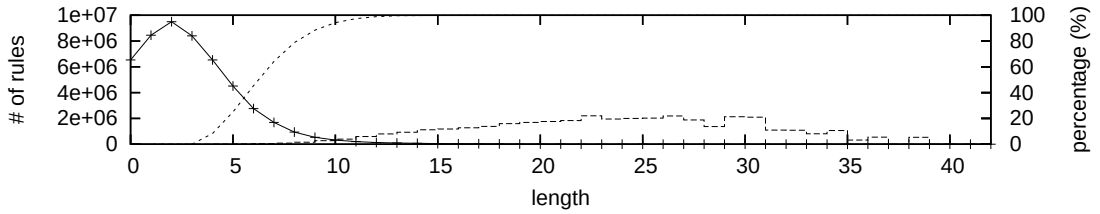


Figure 6: The solid-line curve represents the distribution of all rules against permutation lengths. The dashed-line stairs indicate the percentage of non-binarizable rules in our initial rule set while the dotted-line denotes that percentage among all permutations.

4.1 How many rules are binarizable?

It has been shown by Shapiro and Stephens (1991) and Wu (1997, Sec. 4) that the percentage of binarizable cases over all permutations of length n quickly approaches 0 as n grows (see Figure 6). However, for machine translation, it is more meaningful to compute the ratio of binarizable rules extracted from real text. Our rule set is obtained by first doing word alignment using GIZA++ on a Chinese-English parallel corpus containing 50 million words in English, then parsing the English sentences using a variant of Collins parser, and finally extracting rules using the graph-theoretic algorithm of Galley et al. (2004). We did a "spectrum analysis" on the resulting rule set with 50,879,242 rules. Figure 6 shows how the rules are distributed against their lengths (number of nonterminals). We can see that the percentage of non-binarizable rules in each bucket of the same length does not exceed 25%. Overall, 99.7% of the rules are binarizable. Even for the 0.3% non-binarizable rules, human evaluations show that the majority of them are due to alignment errors. It is also interesting to know that 86.8% of the rules have monotonic permutations, i.e. either taking identical or totally inverted order.

4.2 Does synchronous binarizer help decoding?

We did experiments on our CKY-based decoder with two binarization methods. It is the responsibility of the binarizer to instruct the decoder how to compute the language model scores from children nonterminals in each rule. The baseline method is monolingual left-to-right binarization. As shown in Section 1, decoding complexity with this method is exponential in the size of the longest rule and since we postpone all the language model scorings, pruning in this case is also biased.

system	bleu
monolingual binarization	36.25
synchronous binarization	*38.44*
alignment-template system	37.00

Table 1: Syntax-based systems vs. ATS

To move on to synchronous binarization, we first did an experiment using the above baseline system without the 0.3% non-binarizable rules and did not observe any difference in BLEU scores. So we safely move a step further, focusing on the binarizable rules only.

The decoder now works on the binary translation rules supplied by an external synchronous binarizer. As shown in Section 1, this results in a simplified decoder with a polynomial time complexity, allowing less aggressive and more effective pruning based on both translation model and language model scores.

We compare the two binarization schemes in terms of translation quality with various pruning thresholds. The rule set is that of the previous section. The test set has 116 Chinese sentences of no longer than 15 words. Both systems use trigram as the integrated language model. Figure 7 demonstrates that decoding accuracy is significantly improved after synchronous binarization. The number of edges proposed during decoding is used as a measure of the size of search space, or time efficiency. Our system is consistently faster and more accurate than the baseline system.

We also compare the top result of our synchronous binarization system with the state-of-the-art alignment-template approach (ATS) (Och and Ney, 2004). The results are shown in Table 1. Our system has a promising improvement over the ATS

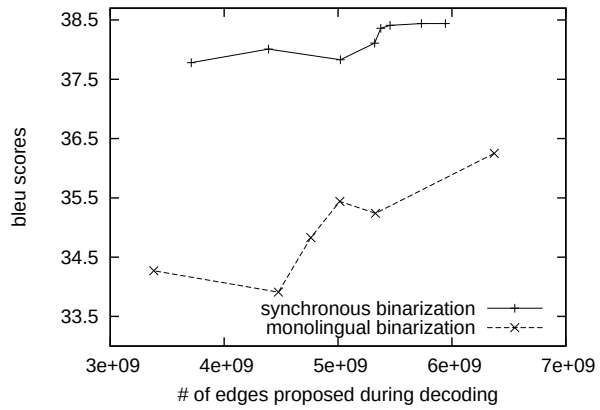


Figure 7: Comparing the two binarization methods in terms of translation quality against search effort.

system which is trained on a larger data-set but tuned independently.

5 Conclusion

Modeling reorderings between languages has been a major challenge for machine translation. This work shows that the majority of syntactic reorderings, at least between languages like English and Chinese, can be efficiently decomposed into hierarchical binary reorderings. From a modeling perspective, on the other hand, it is beneficial to start with a richer representation that has more transformational power than ITG or binary SCFG. Our work shows how to convert it back to a computationally friendly form without harming much of its expressiveness. As a result, decoding with n-gram models can be fast and accurate, making it possible for our syntax-based system to overtake a comparable phrase-based system in BLEU score. We believe that extensions of our technique to more powerful models such as synchronous tree-adjoining grammar (Shieber and Schabes, 1990) is an interesting area for further work.

Acknowledgments Much of this work was done when H. Zhang and L. Huang were visiting USC/ISI. The authors wish to thank Wei Wang, Jonathan Graehl and Steven DeNeefe for help with the experiments. We are also grateful to Daniel Marcu, Giorgio Satta, and Aravind Joshi for discussions. This work was partially supported by NSF ITR IIS-09325646 and NSF ITR IIS-0428020.

References

Albert V. Aho and Jeffery D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL-05*, pages 263–270, Ann Arbor, Michigan.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of ACL-03, companion volume*, Sapporo, Japan.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT/NAACL-04*.

Liang Huang, Hao Zhang, and Daniel Gildea. 2005. Machine translation as lexicalized parsing with hooks. In *Proceedings of IWPT-05*, Vancouver, BC.

Kevin Knight and Jonathan Graehl. 2005. An overview of probabilistic tree transducers for natural language processing. In *Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*. LNCS.

I. Dan Melamed. 2003. Multitext grammars and synchronous parsers. In *Proceedings of NAACL-03*, Edmonton.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4).

Giorgio Satta and Enoch Peserico. 2005. Some computational complexity results for synchronous context-free grammars. In *Proceedings of HLT/EMNLP-05*, pages 803–810, Vancouver, Canada, October.

L. Shapiro and A. B. Stephens. 1991. Bootstrap percolation, the Schröder numbers, and the n-kings problem. *SIAM Journal on Discrete Mathematics*, 4(2):275–280.

Stuart Shieber and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *COLING-90*, volume III, pages 253–258.

Dekai Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *34th Annual Meeting of the Association for Computational Linguistics*.

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

Modelling User Satisfaction and Student Learning in a Spoken Dialogue Tutoring System with Generic, Tutoring, and User Affect Parameters

Kate Forbes-Riley
Learning Research & Development Ctr
University of Pittsburgh
Pittsburgh, PA 15260
forbesk@cs.pitt.edu

Diane J. Litman
Learning Research & Development Ctr
Dept. of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260
litman@cs.pitt.edu

Abstract

We investigate using the PARADISE framework to develop predictive models of system performance in our spoken dialogue tutoring system. We represent performance with two metrics: user satisfaction and student learning. We train and test predictive models of these metrics in our tutoring system corpora. We predict user satisfaction with 2 parameter types: 1) system-generic, and 2) tutoring-specific. To predict student learning, we also use a third type: 3) user affect. Although generic parameters are useful predictors of user satisfaction in other PARADISE applications, overall our parameters produce less useful user satisfaction models in our system. However, generic and tutoring-specific parameters do produce useful models of student learning in our system. User affect parameters can increase the usefulness of these models.

1 Introduction

In recent years the development of *spoken dialogue* tutoring systems has become more prevalent, in an attempt to close the performance gap between human and computer tutors (Mostow and Aist, 2001; Pon-Barry et al., 2004; Litman et al., 2006). *Student learning* is a primary metric for evaluating the performance of these systems; it can be measured, e.g., by comparing student pretests taken prior to system use with posttests taken after system use.

In other types of spoken dialogue systems, the user's subjective judgments about using the system are often considered a primary system performance metric; e.g., user satisfaction has been measured via surveys which ask users to rate systems during use along dimensions such as task ease, speech input/output quality, user expectations and expertise, and user future use (Möller, 2005b; Walker et al., 2002; Bonneau-Maynard et al., 2000; Walker et al., 2000; Shriberg et al., 1992). However, it is expensive to run experiments over large numbers of users to obtain reliable system performance measures.

The PARADISE model (Walker et al., 1997) proposes instead to *predict* system performance, using parameters representing interaction costs and benefits between system and user, including task success, dialogue efficiency, and dialogue quality. More formally, a set of interaction parameters are measured in a spoken dialogue system corpus, then used in a multivariate linear regression to predict the target performance variable. The resulting model is described by the formula below, where there are n interaction parameters, p_i, each weighted by the analysis with a coefficient, w_i, which will be negative or positive, depending on whether the model treats p_i as a cost or benefit, respectively. The model can then be used to estimate performance during system design, with the design goals of minimizing costs and maximizing benefits.

$$\textbf{System Performance} = \sum_{i=1}^{n} \mathrm{w}_i * \mathrm{p}_i$$

We investigate using PARADISE to develop predictive models of performance in our spoken dialogue tutoring system. Although to our knowledge,

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 264–271,
New York, June 2006.

Corpus	Date	Voice	#Dialogues	#Students	#with Survey	#with Tests	#with Affect
SYN03	2003	synthesized	100	20	0	20	20
PR05	2005	pre-recorded	140	28	28	28	17
SYN05	2005	synthesized	145	29	29	29	0

Table 1: Summary of our 3 ITSPOKE Corpora

prior PARADISE applications have only used user satisfaction to represent performance, we hypothesize that other metrics may be more relevant when PARADISE is applied to tasks that are not optimized for user satisfaction, such as our spoken dialogue tutoring system. We thus use 2 metrics to represent performance: 1) a generic metric of user satisfaction computed via user survey, 2) a tutoring-specific metric of student learning computed via student pretest and posttest scores. We train and test predictive models of these metrics on multiple system corpora.

To predict user satisfaction, we use 2 types of interaction parameters: 1) system-generic parameters such as used in other PARADISE applications, e.g. speech recognition performance, and 2) tutoring-specific parameters, e.g. student correctness. To predict student learning, we also use a third type of parameter: 3) manually annotated user affect. Although prior PARADISE applications have tended to use system-generic parameters, we hypothesize that task-specific and user affect parameters may also prove useful. We emphasize that user affect parameters are still system-generic; user affect has been annotated and/or automatically predicted in other types of spoken dialogue systems, e.g. as in (Lee et al., 2002; Ang et al., 2002; Batliner et al., 2003).

Our results show that, although generic parameters were useful predictors of user satisfaction in other PARADISE applications, overall our parameters produce less useful user satisfaction models in our tutoring system. However, generic and tutoring-specific parameters do produce useful models of student learning in our system. Generic user affect parameters increase the usefulness of these models.

2 Spoken Dialogue Tutoring Corpora

ITSPOKE (**I**ntelligent **T**utoring **SPOKE**n dialogue system) (Litman et al., 2006) is a *speech-enabled* tutor built on top of the *text-based* Why2-Atlas conceptual physics tutor (VanLehn et al., 2002). In ITSPOKE, a student first types an essay into a web-based interface answering a qualitative physics problem. ITSPOKE then analyzes the essay and engages the student in spoken dialogue to correct misconceptions and elicit more complete explanations. Student speech is digitized from the microphone input and sent to the Sphinx2 recognizer. Sphinx2's most probable "transcription" is then sent to Why2-Atlas for syntactic, semantic and dialogue analysis. Finally, the text response produced by Why2-Atlas is converted to speech as described below, then played in the student's headphones and displayed on the interface. After the dialogue, the student revises the essay, thereby ending the tutoring or causing another round of tutoring/essay revision.

For this study, we used 3 ITSPOKE corpora, shown in Table 1.[1] The **SYN03** corpus was collected in 20**03** for an evaluation comparing learning in typed and spoken human and computer tutoring (Litman et al., 2006). ITSPOKE's voice was **syn**thesized with the Cepstral text-to-speech system, and its speech recognizer was trained from pilot ITSPOKE studies and Why2-Atlas evaluations. The **PR05** and **SYN05** corpora were collected in 20**05**, to evaluate the impact of tutor voice quality (Forbes-Riley et al., 2006). For these 2 corpora, ITSPOKE used an updated speech recognizer further trained on the **SYN03** corpus. For the **SYN05** corpus, ITSPOKE used the **syn**thesized tutor voice from the **SYN03** corpus; for the **PR05** corpus, ITSPOKE used a **pr**e-recorded tutor voice from a paid voice talent. Figure 1 gives an annotated (Section 3) **PR05** excerpt (**ASR** shows what ITSPOKE heard).

The same experimental procedure was used to collect all 3 ITSPOKE corpora: college students who had taken no college physics: 1) read a small document of background material, 2) took a pretest measuring initial physics knowledge, 3) work through a set of 5 problems (dialogues) with ITSPOKE, 4) took a posttest similar to the pretest.

[1] The user populations of the 2005 corpora and 2003 corpus are different, due to variation in year and recruitment method.

Subjects in the **PR05** and **SYN05** corpora also completed a survey probing user satisfaction after taking the posttest (**SYN03** corpus subjects did not). Our survey, shown in Figure 2, is essentially the same as the one used in the DARPA Communicator multi-site evaluation (Walker et al., 2002). Although tailored lexically for a tutoring system, these statements are generally applicable to spoken dialogue systems. Students rated their degree of agreement with each statement on a scale of 1 to 5.

PHYSICS PROBLEM: Suppose a lightweight car and a massive truck hit a patch of frictionless ice and have a head-on collision. Upon which vehicle is the impact force greater? Which vehicle undergoes the greater change in its motion? Defend your answers.

ESSAY UNDER DISCUSSION: The truck exerts an equal and opposite reaction force on the car. The car has a greater acceleration. The car will have a greater change in its motion.

. . . excerpt at 28.8 minutes into dialogue. . .

TUTOR$_{60}$: So, when you add forces together, besides the magnitudes of the forces, what must you also consider?

STUDENT$_{57}$: direct- direction (**ASR:** the truck in direction) *[correct, uncertain, non-frustrated/angry]*

TUTOR$_{61}$: Yes. [...] The vector sum of the forces acting on the body is also equal to the mass of the body multiplied by its acceleration. What is another term for the vector sum of forces acting on a body?

STUDENT$_{58}$: net force (**ASR:** net force) *[correct, neutral, non-frustrated/angry]*

TUTOR$_{62}$: That's super. Another term for the resultant force is the net force. In this relationship, what is the direction of the net force applied to a body and the direction of the body's acceleration?

STUDENT$_{59}$: same same same (**ASR:** seems same type seems) *[correct, certain, frustrated/angry]*

Figure 1: Annotated **PR05** Corpus Excerpt

Task Ease: It was easy to learn from the tutor.
TTS Performance: It was easy to understand the tutor.
User Expertise: I knew what I could say or do at each point in the conversations with the tutor.
Expected Behavior: The tutor worked the way I expected it to.
Future Use: Based on my experience using the tutor to learn physics, I would like to use such a tutor regularly.

ALMOST ALWAYS (5), OFTEN (4), SOMETIMES (3), RARELY (2), ALMOST NEVER (1)

Figure 2: ITSPOKE Survey Questionnaire

3 Interaction Parameters

3.1 Dialogue System-Generic Parameters

Prior PARADISE applications predicted user satisfaction using a wide range of system-generic parameters, which include measures of speech recognition quality (e.g. word error rate), measures of dialogue communication and efficiency (e.g. total turns and elapsed time), and measures of task completion (e.g. a binary representation of whether the task was completed) (Möller, 2005a; Möller, 2005b; Walker et al., 2002; Bonneau-Maynard et al., 2000; Walker et al., 2000; Walker et al., 1997). In this prior work, each dialogue between user and system represents a single "task" (e.g., booking airline travel), thus these measures are calculated on a per-dialogue basis.

In our work, the entire tutoring session represents a single "task", and every student in our corpora completed this task. Thus we extract 13 system-generic parameters on a per-student basis, i.e. over the 5 dialogues for each user, yielding a single parameter value for each student in our 3 corpora.

First, we extracted 9 parameters representing dialogue communication and efficiency. Of these parameters, 7 were used in prior PARADISE applications: Time on Task, Total ITSPOKE Turns and Words, Total User Turns and Words, Average ITSPOKE Words/Turn, and Average User Words/Turn. Our 2 additional "communication-related" (Möller, 2005a) parameters measure system-user interactivity, but were not used in prior work (to our knowledge): Ratio of User Words to ITSPOKE Words, Ratio of User Turns to ITSPOKE Turns.

Second, we extracted 4 parameters representing speech recognition quality, which have also been used in prior work: Word Error Rate, Concept Accuracy, Total Timeouts, Total Rejections[2].

3.2 Tutoring-Specific Parameters

Although prior PARADISE applications tend to use system-generic parameters, we hypothesize that task-specific parameters may also prove useful for predicting performance. We extract 12 *tutoring-specific* parameters over the 5 dialogues for each student, yielding a single parameter value per student, for each student in our 3 corpora. Although these parameters are specific to our tutoring system, similar parameters are available in other tutoring systems.

First, we hypothesize that the *correctness* of the students' turns with respect to the tutoring topic

[2]A Timeout occurs when ITSPOKE does not hear speech by a pre-specified time interval. A Rejection occurs when ITSPOKE's confidence score for its ASR output is too low.

(physics, in our case) may play a role in predicting system performance. Each of our student turns is automatically labeled with 1 of 3 "Correctness" labels by the ITSPOKE semantic understanding component: *Correct, Incorrect, Partially Correct*. Labeled examples are shown in Figure 1. From these 3 Correctness labels, we derive 9 parameters: a Total and a Percent for each label, and a Ratio of each label to every other label (e.g. Correct/Incorrect).

Second, students write and then may modify their physics essay at least once during each dialogue with ITSPOKE. We thus hypothesize that like "Correctness", the total number of essays per student may play a role in predicting system performance.

Finally, although student test scores before/after using ITSPOKE will be used as our student learning metric, we hypothesize that these scores may also play a role in predicting user satisfaction.

3.3 User Affect Parameters

We hypothesize that user affect plays a role in predicting user satisfaction and student learning. Although affect parameters have not been used in other PARADISE studies (to our knowledge), they are generic; for example, in various spoken dialogue systems, user affect has been annotated and automatically predicted from e.g., acoustic-prosodic and lexical features (Litman and Forbes-Riley, 2004b; Lee et al., 2002; Ang et al., 2002; Batliner et al., 2003).

As part of a larger investigation into emotion *adaptation*, we are manually annotating the student turns in our corpora for affective state. Currently, we are labeling 1 of 4 states of "Certainness": *certain, uncertain, neutral, mixed (certain and uncertain)*, and we are separately labeling 1 of 2 states of "Frustration/Anger": *frustrated/angry, non-frustrated/angry*. These affective states[3] were found in pilot studies to be most prevalent in our tutoring dialogues[4], and are also of interest in other dialogue research, e.g. tutoring (Bhatt et al., 2004; Moore et al., 2004; Pon-Barry et al., 2004) and spoken dialogue (Ang et al., 2002). Labeled examples are shown in Figure 1.[5] To date, one paid annotator has labeled all student turns in our **SYN03** corpus, and all the turns of 17 students in our **PR05** corpus.[6]

From these labels, we derived 25 **User Affect** parameters per student, over the 5 dialogues for that student. First, for each Certainness label, we computed a Total, a Percent, and a Ratio to each other label. We also computed a Total for each *sequence* of identical Certainness labels (e.g. Certain:Certain), hypothesizing that states maintained over multiple turns may have more impact on performance than single occurrences. Second, we computed the same parameters for each Frustration/Anger label.

4 Prediction Models

In this section, we first investigate the usefulness of our system-generic and tutoring-specific parameters for training models of user satisfaction and student learning in our tutoring corpora with the PARADISE framework. We use the SPSS statistical package with a stepwise multivariate linear regression procedure[7] to automatically determine parameter inclusion in the model. We then investigate how well these models generalize across different user-system configurations, by testing the models in different corpora and corpus subsets. Finally, we investigate whether generic user affect parameters increase the usefulness of our student learning models.

4.1 Prediction Models of User Satisfaction

Only subjects in the **PR05** and **SYN05** corpora completed a user survey (Table 1). Each student's responses were summed to yield a single user satisfaction total per student, ranging from 9 to 24 across corpora (the possible range is 5 to 25), with no difference between corpora (p = .46). This total was used as our user satisfaction metric, as in (Möller, 2005b; Walker et al., 2002; Walker et al., 2000).[8]

[3] We use "affect" and "affective state" loosely to cover student emotions and attitudes believed to be relevant for tutoring.

[4] For a full list of affective states identified in these pilot studies, see (Litman and Forbes-Riley, 2004a).

[5] Annotations were performed from both audio and transcription within a speech processing tool.

[6] In a preliminary agreement study, a second annotator labeled the entire **SYN03** corpus for *uncertain* versus *other*, yielding 90% inter-annotator agreement (0.68 Kappa).

[7] At each step, the parameter with the highest partial correlation with the target predicted variable, controlled for all previously entered parameters, is entered in the equation, until the remaining parameters do not increase R^2 by a significant amount or do not yield a significant model.

[8] Researchers have also used average score (Möller, 2005b; Walker et al., 1997); single survey statements can also be used (Walker et al., 1997). We tried these variations, and our R^2 results were similar, indicating robustness across variations.

Training Data	R^2	Predictors	Testing Data	R^2
PR05	.274	INCORRECTS, ESSAYS	SYN05	.001
SYN05	.068	TUT WDS/TRN	PR05	.018
PR05:half1	.335	PARTCORS/INCORS	PR05:half2	.137
PR05:half2	.443	STU TRNS	PR05:half1	.079
SYN05:half1	.455	STU TRNS/TUT TRNS	SYN05:half2	.051
SYN05:half2	.685	TUT WDS/TRN, STU WDS/TRN, CORRECTS	SYN05:half1	.227

Table 2: Testing the Predictive Power of User Satisfaction Models

We trained a user satisfaction model on each corpus, then tested it on the other corpus. In addition, we split each corpus in half randomly, then trained a user satisfaction model on each half, and tested it on the other half. We hypothesized that despite the decrease in the dataset size, models trained and tested in the same corpus would have higher generalizability than models trained on one corpus and tested on the other, due to the increased data homogeneity within each corpus, since each corpus used a different ITSPOKE version. As predictors, we used only the 13 system-generic and 12 tutoring-specific parameters that were available for all subjects.

Results are shown in Table 2. The first and fourth columns show the training and test data, respectively. The second and fifth columns show the user satisfaction variance accounted for by the trained model in the training and test data, respectively. The third column shows the parameters that were selected as predictors of user satisfaction in the trained model, ordered by degree of contribution[9].

For example, as shown in the first row, the model trained on the **PR05** corpus uses Total Incorrect student turns as the strongest predictor of user satisfaction, followed by Total Essays; these parameters are not highly correlated[10]. This model accounts for 27.4% of the user satisfaction variance in the **PR05** corpus. When tested on the **SYN05** corpus, it accounts for 0.1% of the user satisfaction variance.

The low R^2 values for both training and testing in the first two rows show that neither corpus yields a very powerful model of user satisfaction even in the training corpus, and this model does not generalize very well to the test corpus. As hypothesized, training and testing in a single corpus yields higher R^2 values for testing, as shown in the last four rows, although these models still account for less than a quarter of the variance in the test data. The increased R^2 values for training here may indicate over-fitting. Across all 6 experiments, there is almost no overlap of parameters used to predict user satisfaction.

Overall, these results show that this method of developing an ITSPOKE user satisfaction model is very sensitive to changes in training data; this was also found in other PARADISE applications (Möller, 2005b; Walker et al., 2000). Some applications have also reported similarly low R^2 values for *testing* both within a corpus (Möller, 2005b) and also when a model trained on one system corpus is tested on another system corpus (Walker et al., 2000). However, most PARADISE applications have yielded higher R^2 values than ours for *training* (Möller, 2005b; Walker et al., 2002; Bonneau-Maynard et al., 2000; Walker et al., 2000).

We hypothesize two reasons for why our experiments did not yield more useful user satisfaction models. First, in prior PARADISE applications, users completed a survey after *every* dialogue with the system. In our case, subjects completed only one survey, at the end of the experiment (5 dialogues). It may be that this "per-student" unit for user satisfaction is too large to yield a very powerful model; i.e., this measure is not fine-grained enough. In addition, tutoring systems are not designed to maximize user satisfaction, but rather, their design goal is to maximize student learning. Moreover, prior tutoring studies have shown that certain features correlated with student learning do not have the same relationship to user satisfaction (e.g. are not predictive

[9]The ordering reflects the standardized coefficients (beta weights), which are computed in SPSS based on scaling of the input parameters, to enable an assessment of the predictive power of each parameter relative to the others in a model.

[10]Hereafter, predictors in a model are not highly correlated ($R \geq .70$) unless noted. Linear regression does not assume that predictors are independent, only that they are not highly correlated. Because correlations above R =.70 can affect the coefficients, deletion of redundant predictors may be advisable.

Training Data	R^2	Predictors	Testing Data	R^2
PR05	.556	PRE, %CORRECT	SYN05	.636
SYN05	.736	PRE, INCORS/CORS, STU WDS/TRN	PR05	.472
PR05:half1	.840	PRE, PARTCORRECTS	PR05:half2	.128
PR05:half2	.575	PARTCORS/INCORS, PRE	PR05:half1	.485
SYN05:half1	.580	PRE, STU WDS/TRN	SYN05:half2	.556
SYN05:half2	.855	PRE, TIMEOUTS	SYN05:half1	.384
PR05+SYN03	.413	PRE, TIME	SYN05	.586
PR05+SYN05	.621	PRE, INCORS/CORS	SYN03	.237
SYN05+SYN03	.590	INCORS/CORS, %INCORRECT, PRE, TIME	PR05	.244

Table 3: Testing the Predictive Power of Student Learning Models with the Same Datasets

or have an opposite relationship) (Pon-Barry et al., 2004). In fact, it may be that user satisfaction is not a metric of primary relevance in our application.

4.2 Prediction Models of Student Learning

As in other tutoring research, e.g. (Chi et al., 2001; Litman et al., 2006), we use posttest score (POST) controlled for pretest score (PRE) as our target student learning prediction metric, such that POST is our target variable and PRE is always a parameter in the final model, although it is not necessarily the strongest predictor.[11] In this way, we measure student learning *gains*, not just final test score.

As shown in Table 1, all subjects in our 3 corpora took the pretest and posttest. However, in order to compare our student learning models with our user satisfaction models, our first experiments predicting student learning used the same training and testing datasets that were used to predict user satisfaction in Section 4.1 (i.e. we ran the same experiments except we predicted POST controlled for PRE instead of user satisfaction). Results are shown in the first 6 rows of Table 3.

As shown, these 6 models all account for more than 50% of the POST variance in the training data. Furthermore, most of them account for close to, or more than, 50% of the POST variance in the test data. Although again we hypothesized that training and testing in one corpus would yield higher R^2 values for testing, this is not consistently the case; two of these models had the highest R^2 values for training and the lowest R^2 values for testing (**PR05:half1** and **SYN05:half2**), suggesting over-fitting.

Overall, these results show that this is an effective method of developing a prediction model of student learning for ITSPOKE, and is less sensitive to changes in training data than it was for user satisfaction. Moreover, there is more overlap in these 6 models of parameters that are useful for predicting student learning (besides PRE); "Correctness" parameters and dialogue communication and efficiency parameters appear to be most useful overall.

Our next 3 experiments investigated how our student learning models are impacted by including our third **SYN03** corpus. Using the same 25 parameters, we trained a learning model on each set of two combined corpora, then tested it on the other corpus. Results are shown in the last 3 rows of Table 3.

As shown, these models still account for close to, or more than, 50% of the student learning variance in the training data.[12] The model trained on **PR05+SYN03** accounts for the most student learning variance in the test data, showing that the training data that is most similar to the test data will yield the highest generalizability. That is, the combined **PR05+SYN03** corpora contains subjects drawn from the same subject pool (2005) as the **SYN05** test data, and also contains subjects who interacted with the same tutor voice (synthesized) as this test data. In contrast, the combined **PR05+SYN05** corpora did not overlap in user population with the **SYN03** test data, and the combined **SYN05+SYN03** corpora did not share a tutor voice with the **PR05** test data. "Correctness" parameters

[11] In SPSS, we regress two independent variable blocks. The first block contains PRE, which is regressed with POST using the "enter" method, forcing inclusion of PRE in the final model. The second block contains all remaining independent variables, which are regressed using the stepwise method.

[12] However, INCORS/CORS and %INCORRECT are highly correlated in the **SYN05+SYN03** model, showing redundancy.

Training Data	R^2	Predictors	Testing Data	R^2
SYN03 (affect)	.644	TIME, PRE, NEUTRAL	PR05:17	.411
PR05:17 (affect)	.835	PRE, NFA:NFA, STU WDS/TRN	SYN03	.127
SYN03	.478	PRE, TIME	PR05:17	.340
PR05:17	.609	PRE, STU TRNS/TUT TRNS	SYN03	.164

Table 4: Testing the Predictive Power of Student Learning Models with User Affect Parameters

and dialogue communication and efficiency parameters are consistently used as predictors in all 9 of these student learning models.

4.3 Adding User Affect Parameters

Our final experiments investigated whether our 25 user affect parameters impacted the usefulness of the student learning models. As shown in Table 1, all 20 subjects in our **SYN03** corpus were annotated for user affect, and 17 subjects in our **PR05** corpus were annotated for user affect. We trained a model of student learning on each of these datasets, then tested it on the other dataset.[13] As predictors, we included our 25 user affect parameters along with the 13 system-generic and 12 tutoring-specific interaction parameters. These results are shown in the first two rows of Table 4. We also reran these experiments without user affect parameters, to gauge the impact of the user affect parameters. These results are shown in the last two rows of Table 4. We hypothesized that user affect parameters would produce more useful models, because prior tutoring research has shown correlations between user affect and student learning (e.g. (Craig et al., 2004)).

As shown in the first two rows, user affect predictors appear in both models where these parameters were included. The models trained on **SYN03** use pretest score and Total Time on Task as predictors; when affect parameters are included, "Neutral Certainness" is added as a predictor, which increases the R^2 values for both training and testing. However, the two models trained on **PR05:17** show no predictor overlap (besides PRE). Moreover, the **PR05:17** model that includes an affect predictor (Total Sequence of 2 Non-Frustrated/Angry turns) has the highest training R^2, but the lowest testing R^2 value.

[13] As only 17 subjects have both user affect annotation and user surveys, there is not enough data currently to train and test a user satisfaction model including user affect parameters.

5 Conclusions and Current Directions

Prior work in the tutoring community has focused on correlations of single features with learning; our results suggest that PARADISE is an effective method of extending these analyses. For the dialogue community, our results suggest that as spoken dialogue systems move into new applications not optimized for user satisfaction, such as tutoring systems, other measures of performance may be more relevant, and generic user affect parameters may be useful.

Our experiments used many of the same system-generic parameters as prior studies, and some of these parameters predicted user satisfaction both in our models and in prior studies' models (e.g., system words/turn (Walker et al., 2002)). Nonetheless, overall our user satisfaction models were not very powerful even for training, were sensitive to training data changes, showed little predictor overlap, and did not generalize well to test data. Our user satisfaction metric may not be fine-grained enough; in other PARADISE studies, users took a survey after *every* dialogue with the system. In addition, tutoring systems are not designed to maximize user satisfaction; their goal is to maximize student learning.

Our student learning models were much more powerful and less sensitive to changes in training data. Our best models explained over 50% of the student learning variance for training and testing, and both student "Correctness" parameters and dialogue communication and efficiency parameters were often useful predictors. User affect parameters further improved the predictive power of one student learning model for both training and testing.

Once our user affect annotations are complete, we can further investigate their use to predict student learning and user satisfaction. Unlike our other parameters, these annotations are not currently available, although they can be predicted automatically (Litman and Forbes-Riley, 2004b), in our sys-

tem. However, as in (Batliner et al., 2003), our prior work suggests that linguistic features reflective of affective states can replace affect annotation (Forbes-Riley and Litman, 2005). In future work we will use such features in our prediction models. Finally, we are also annotating tutor and student dialogue acts and automating the tutor act annotations; when complete we can investigate their usefulness in our prediction models; dialogue acts have also been used in prior PARADISE applications (Möller, 2005a).

Acknowledgements

NSF (0325034 & 0328431) supports this research. We thank Pam Jordan and the NLP Group.

References

J. Ang, R. Dhillon, A. Krupski, E.Shriberg, and A. Stolcke. 2002. Prosody-based automatic detection of annoyance and frustration in human-computer dialog. In *Proc. Int. Conf. Spoken Language Processing (ICSLP)*.

A. Batliner, K. Fischer, R. Huber, J. Spilker, and E. Noth. 2003. How to find trouble in communication. *Speech Communication*, 40:117–143.

K. Bhatt, M. Evens, and S. Argamon. 2004. Hedged responses and expressions of affect in human/human and human/computer tutorial interactions. In *Proc. 26th Annual Meeting of the Cognitive Science Society*.

H. Bonneau-Maynard, L. Devillers, and S. Rosset. 2000. Predictive performance of dialog systems. In *Proc. Language Resources and Evaluation Conf. (LREC))*.

M. T. H. Chi, S. A. Siler, H. Jeong, T. Yamauchi, and R. G. Hausmann. 2001. Learning from human tutoring. *Cognitive Science*, 25:471–533.

S. Craig, A. Graesser, J. Sullins, and B. Gholson. 2004. Affect and learning: An exploratory look into the role of affect in learning. *Journal of Educational Media*, 29:241–250.

K. Forbes-Riley and D. Litman. 2005. Correlating student acoustic-prosodic profiles with student learning in spoken tutoring dialogues. In *Proc. INTERSPEECH*.

K. Forbes-Riley, D. Litman, S. Silliman, and J. Tetreault. 2006. Comparing synthesized versus pre-recorded tutor speech in an intelligent tutoring spoken dialogue system. In *Proc. FLAIRS*.

C.M. Lee, S. Narayanan, and R. Pieraccini. 2002. Combining acoustic and language information for emotion recognition. In *Proc. ICSLP*.

D. Litman and K. Forbes-Riley. 2004a. Annotating student emotional states in spoken tutoring dialogues. In *Proc. SIGdial*, pages 144–153.

D. Litman and K. Forbes-Riley. 2004b. Predicting student emotions in computer-human tutoring dialogues. In *Proc. ACL*, pages 352–359.

D. Litman, C. Rosé, K. Forbes-Riley, K. VanLehn, D. Bhembe, and S. Silliman. 2006. Spoken versus typed human and computer dialogue tutoring. *Intnl Jnl of Artificial Intelligence in Education, To Appear*.

S. Möller. 2005a. Parameters for quantifying the interactioin with spoken dialogue telephone services. In *Proc. SIGdial*.

S. Möller. 2005b. Towards generic quality prediction models for spoken dialogue systems - a case study. In *Proc. INTERSPEECH*.

J. D. Moore, K. Porayska-Pomsta, S. Varges, and C. Zinn. 2004. Generating tutorial feedback with affect. In *Proc. FLAIRS*.

J. Mostow and G. Aist. 2001. Evaluating tutors that listen: An overview of Project LISTEN. In K. Forbus and P. Feltovich, editors, *Smart Machines in Education*.

H. Pon-Barry, B. Clark, E. Owen Bratt, K. Schultz, and S. Peters. 2004. Evaluating the effectiveness of SCoT: a Spoken Conversational Tutor. In *Proc. of ITS 2004 Workshop on Dialogue-based Intelligent Tutoring Systems: State of the Art and New Research Directions*.

E. Shriberg, E. Wade, and P. Price. 1992. Human-machine problem solving using spoken language systems (SLS): Factors affecting performance and user satisfaction. In *Proc. DARPA Speech and NL Workshop*, pages 49–54.

K. VanLehn, P. W. Jordan, C. P. Rosé, D. Bhembe, M. Böttner, A. Gaydos, M. Makatchev, U. Pappuswamy, M. Ringenberg, A. Roque, S. Siler, R. Srivastava, and R. Wilson. 2002. The architecture of Why2-Atlas: A coach for qualitative physics essay writing. In *Proc. Intelligent Tutoring Systems*.

M. Walker, , D. Litman, C. Kamm, and A. Abella. 1997. PARADISE: A framework for evaluating spoken dialogue agents. In *Proc. ACL/EACL*, pages 271–280.

M. Walker, C. Kamm, and D. Litman. 2000. Towards developing general models of usability with PARADISE. *Natural Language Engineering*, 6:363–377.

M. Walker, A. Rudnicky, R. Prasad, J. Aberdeen, E. Bratt, J. Garofolo, H. Hastie, A. Le, B. Pellom, A. Potamianos, R. Passonneau, S. Roukos, G. Sanders, S. Seneff, and D. Stallard. 2002. DARPA communicator: Cross-system results for the 2001 evaluation. In *Proc. Internat. Conf. on Spoken Language Processing (ICSLP)*.

Comparing the Utility of State Features in Spoken Dialogue Using Reinforcement Learning

Joel R. Tetreault
University of Pittsburgh
Learning Research and Development Center
Pittsburgh PA, 15260, USA
tetreaul@pitt.edu

Diane J. Litman
University of Pittsburgh
Department of Computer Science
Learning Research and Development Center
Pittsburgh PA, 15260, USA
litman@cs.pitt.edu

Abstract

Recent work in designing spoken dialogue systems has focused on using Reinforcement Learning to automatically learn the best action for a system to take at any point in the dialogue to maximize dialogue success. While policy development is very important, choosing the best features to model the user state is equally important since it impacts the actions a system should make. In this paper, we compare the relative utility of adding three features to a model of user state in the domain of a spoken dialogue tutoring system. In addition, we also look at the effects of these features on what type of a question a tutoring system should ask at any state and compare it with our previous work on using feedback as the system action.

1 Introduction

A host of issues confront spoken dialogue system designers, such as choosing the best system action to perform given any user state, and also selecting the right features to best represent the user state. While recent work has focused on using Reinforcement Learning (RL) to address the first issue (such as (Walker, 2000), (Henderson et al., 2005), (Williams et al., 2005a)), there has been very little empirical work on the issue of feature selection in prior RL approaches to dialogue systems. In this paper, we use a corpus of dialogues of humans interacting with a spoken dialogue tutoring system to show the comparative utility of adding the three features of *concept repetition*, *frustration level*, and *student performance*. These features are not just unique to the tutoring domain but are important to dialogue systems in general. Our empirical results show that these features all lead to changes in what action the system should take, with concept repetition and frustration having the largest effects.

This paper extends our previous work (Tetreault and Litman, 2006) which first presented a methodology for exploring whether adding more complex features to a representation of student state will beneficially alter tutor actions with respect to feedback. Here we present an empirical method of comparing the effects of each feature while also generalizing our findings to a different action choice of what type of follow-up question should a tutor ask the student (as opposed to what type of feedback should the tutor give). In complex domains such as tutoring, testing different policies with real or simulated students can be time consuming and costly so it is important to properly choose the best features before testing, which this work allows us to do. This in turn aids our long-term goal of improving a spoken dialogue system that can effectively adapt to a student to maximize their learning.

2 Background

We follow past lines of research (such as (Levin and Pieraccini, 1997) and (Singh et al., 1999)) for describing a dialogue d as a trajectory within a Markov Decision Process (MDP) (Sutton and Barto, 1998).

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 272–279,
New York, June 2006.

A MDP has four main components: 1: *states* s, 2: *actions* A, 3: a *policy* π, which specifies what is the best action to take in a state, and 4: a *reward function* R which specifies the worth of the entire process. Dialogue management is easily described using a MDP because one can consider the actions as actions made by the system, the state as the dialogue context (which can be viewed as a vector of features, such as ASR confidence or dialogue act), and a reward which for many dialogue systems tends to be task completion success or dialogue length.

Another advantage of using MDP's to model a dialogue space, besides the fact that the primary MDP parameters easily map to dialogue parameters, is the notion of delayed reward. In a MDP, since rewards are often not given until the final states, dynamic programming is used to propagate the rewards back to the internal states to weight the value of each state (called the V-value), as well as to develop an optimal policy π for each state of the MDP. This propagation of reward is done using the *policy iteration* algorithm (Sutton and Barto, 1998) which iteratively updates the V-value and best action for each state based on the values of its neighboring states.

The V-value of each state is important for our purposes not only because it describes the relative worth of a state within the MDP, but as more data is added when building the MDP, the V-values should stabilize, and thus the policies stabilize as well. Since, in this paper, we are comparing policies in a fixed data set it is important to show that the policies are indeed reliable, and not fluctuating.

For this study, we used the MDP infrastructure designed in our previous work which allows the user to easily set state, action, and reward parameters. It then performs policy iteration to generate a policy and V-values for each state. In the following sections, we discuss our corpus, methodology, and results.

3 Corpus

For our study, we used an annotated corpus of 20 human-computer spoken dialogue tutoring sessions (for our work we use the ITSPOKE system (Litman and Silliman, 2004) which uses the text-based Why2-ATLAS dialogue tutoring system as its "back-end" (VanLehn et al., 2002)). The content of the system, and all possible dialogue paths, were authored by physics experts. Each session consists of an interaction with one student over 5 different college-level physics problems, for a total of 100 dialogues. Before each session, the student is asked to read physics material for 30 minutes and then take a pretest based on that material. Each problem begins with the student writing out a short essay response to the question posed by the computer tutor. The fully-automated system assesses the essay for potential flaws in the reasoning and then starts a dialogue with the student, asking questions to help the student understand the confused concepts. The tutor's response and next question is based only on the correctness of the student's last answer. Informally, the dialogue follows a question-answer format. Once the student has successfully completed the dialogue section, he is asked to correct the initial essay. Each of the dialogues takes on average 20 minutes and 60 turns. Finally, the student is given a posttest similar to the pretest, from which we can calculate their normalized learning gain: $NLG = \frac{posttest - pretest}{1 - pretest}$.

State Feature	Values
Certainty	Certain (cer) Uncertain (unc) Neutral (neu)
Frustration	Frustrated (F) Neutral (N),
Correctness	Correct (C) Partially Correct (PC) Incorrect (I)
Percent Correct	50-100% (H)igh 0-49% (L)ow
Concept Repetition	Concept is new (0) Concept is repeated (R)

Table 1: Potential Student State Features in MDP

Prior to our study, the corpus was annotated for Tutor Moves, which can be viewed as Dialogue Acts (Forbes-Riley et al., 2005) [1] and consisted of Tutor Feedback, Question and State Acts. In this corpus, a turn can consist of multiple utterances and thus can be labeled with multiple moves. For example, a tutor can give positive feedback and then ask a question in the same turn. What type of question to ask will be the action choice addressed in this paper.

As for features to include in the student state, we annotated five features as shown in Table 1. Two

[1] The Dialogue Act annotation had a Kappa of 0.67.

Action	Example Turn
SAQ	"Good. What is the direction of that force relative to your fi st?"
CAQ	"What is the defi nition of Newton's Second Law?"
Mix	"Good. If it doesn't hit the center of the pool what do you know about the magnitude of its displacement from the center of the pool when it lands? Can it be zero? Can it be nonzero?"
NoQ	"So you can compare it to my response..."

Table 2: Tutor Actions for MDP

emotion related features, certainty and frustration, were annotated manually prior to this study (Forbes-Riley and Litman, 2005) [2]. Certainty describes how confident a student seemed to be in his answer, while frustration describes how frustrated the student seemed to be when he responded. We include three other automatically extracted features for the Student state: (1) Correctness: whether the student was correct or not; (2) Percent Correct: percentage of correctly answered questions so far for the current problem; (3) Concept Repetition: whether the system is forced to cover a concept again which reflects an area of difficulty for the student.

4 Experimental Method

The goal of this study is to quantify the utility of adding a feature to a baseline state space. We use the following four step process: (1) establish an action set and reward function to be used as constants throughout the test since the state space is the one MDP parameter that will be changed during the tests; (2) establish a baseline state and policy, and (3) add a new feature to that state and test if adding the feature results in policy changes. Every time we create a new state, we make sure that the generated V-values converge. Finally, (4), we evaluate the effects of adding a new feature by using three metrics: (1) number of policy changes (diffs), (2) % policy change, and (3) Expected Cumulative Reward. These three metrics are discussed in more detail in Section 5.2. In this section we focus on the first three steps of the methodology.

4.1 Establishing Actions and Rewards

We use questions as our system action A in our MDP. The action size is 4 (tutor can ask a simple answer question (SAQ), a complex answer question (CAQ), or a combination of the two (Mix), or not ask a question (NoQ)). Examples from our corpus can be seen in Table 2. We selected this as the action because what type of question a tutor should ask is of great interest to the Intelligent Tutoring Systems community, and it generalizes to dialogue systems since asking users questions of varying complexity can elicit different responses.

For the dialogue reward function R we did a median split on the 20 students based on their normalized learning gain, which is a standard evaluation metric in the Intelligent Tutoring Systems community. So 10 students and their respective 5 dialogues were assigned a positive reward of 100 (high learners), and the other 10 students and their respective 5 dialogues were assigned a negative reward of -100 (low learners). The final student turns in each dialogue were marked as either a positive final state (for a high learner) or a negative final state (for a low learner). The final states allow us to propagate the reward back to the internal states. Since no action is taken from the final states, their V-values remain the same throughout policy iteration.

4.2 Establishing a Baseline State and Policy

Currently, our tutoring system's response to a student depends only on whether or not the student answered the last question correctly, so we use correctness as the sole feature in our baseline dialogue state. A student can either be correct, partially correct, or incorrect. Since partially correct responses occur infrequently compared to the other two, we reduced the state size to two by combining Incorrect and Partially Correct into one state (IPC) and keeping Correct (C).

With the actions, reward function, and baseline state all established, we use our MDP tool to generate a policy for both states (see Table 3). The second column shows the states, the third, the policy determined by our MDP toolkit (i.e. the optimal action to

[2] In a preliminary agreement study, a second annotator labeled the entire corpus for *uncertain* versus *other*, yielding 90% inter-annotator agreement (0.68 Kappa).

take in that state with respect to the final reward) and finally how many times the state occurs in our data (state size). So if a student is correct, the best action is to give something other than a question immediately, such as feedback. If the student is incorrect, the best policy is to ask a combination of short and complex answer questions.

#	State	Policy	State Size
1	C	NoQ	1308
2	IPC	Mix	872

Table 3: Baseline Policy

The next step in our experiment is to test whether the policies generated are indeed reliable. Normally, the best way to verify a policy is to conduct experiments and see if the new policy leads to a higher reward for new dialogues. In our context, this would entail running more subjects with the augmented dialogue manager, which could take months. So, instead we check if the polices and values for each state are indeed converging as we add data to our MDP model. The intuition here is that if both of those parameters were varying between a corpus of 19 students versus one of 20 students, then we can't assume that our policy is stable, and hence is not reliable.

To test this out, we made 20 random orderings of our students to prevent any one ordering from giving a false convergence. Each ordering was then passed to our MDP infrastructure such that we started with a corpus of just the first student of the ordering and then determined a MDP policy for that cut, then incrementally added one student at a time until we had added all 20 students. So at the end, 20 random orderings with 20 cuts each provides 400 MDP trials. Finally, we average each cut across the 20 random orderings. The first graph in Figure 1 shows a plot of the average V-values against a cut. The state with the plusses is the positive final state, and the one at the bottom is the negative final state. However, we are most concerned with how the non-final states converge. The plot shows that the V-values are fairly stable after a few initial cuts, and we also verified that the policies remained stable over the 20 students as well (see our prior work (Tetreault and Litman, 2006) for details of this method). Thus we can be sure that our baseline policy is indeed reliable for our corpus.

5 Results

In this section, we investigate whether adding more information to our student state will lead to interesting policy changes. First, we add certainty to our baseline of correctness because prior work (such as (Bhatt et al., 2004), (Liscombe et al., 2005) and (Forbes-Riley and Litman, 2005)) has shown the importance of considering certainty in tutoring systems. We then compare this new baseline's policy (henceforth Baseline 2) with the policies generated when frustration, concept repetition, and percent correctness are included.

We'll first discuss the new baseline state. There are three types of certainty: certain (cer), uncertain (unc), and neutral (neu). Adding these to our state representation increases state size from 2 to 6. The new policy is shown in Table 4. The second and third columns show the original baseline states and their policies. The next column shows the new policy when splitting the original state into the three new states based on certainty (with the policies that differ from the baseline shown in bold). The final column shows the size of each new state. So the first row indicates that if the student is correct and certain, one should give a combination of a complex and short answer question; if the student is correct and neutral, just ask a SAQ; and else if the student is correct and uncertain, give a Mix. The overall trend of adding the certainty feature is that if the student exhibits some emotion (either they are certain or uncertain), the best response is Mix, but for neutral do something else.

#	State	Baseline	Baseline 2	B2 State Size
1	C	NoQ	certain:C **Mix**	663
			neutral:C **SAQ**	480
			uncertain:C **Mix**	165
2	IPC	Mix	certain:IPC Mix	251
			neutral:IPC **NoQ**	377
			uncertain:IPC Mix	244

Table 4: Baseline 2 Policy

We assume that if a feature is important to include in a state representation it should change the policies of the old states. For example, if certainty did not impact how well students learned (as deemed by the MDP) then the policies for certainty, uncertainty,

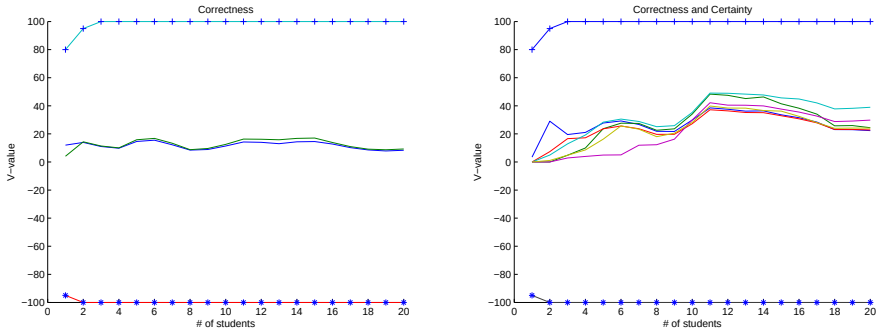


Figure 1: Baseline 1 and 2 Convergence Plots

and neutral would be the same as the original policy for Correct (C) or Incorrect (IPC). However, the figures show otherwise. When certainty is added to the state, only two new states (incorrect while being certain or uncertain) retain the old policy of having the tutor give a mix of SAQ and CAQ. The right graph in Figure 1 shows that for Baseline 2, V-values tend to converge around 10 cuts.

Next, we add Concept Repetition, Frustration, and Percent Correct features individually to Baseline 2. For each of the three features we repeated the reliability check of plotting the V-value convergence and found that the graphs showed convergence around 15 students.

5.1 Feature Addition Results

Policies for the three new features are shown in Table 5 with the policies that differ from Baseline 2's shown in bold. The numbers in parentheses refer to the size of the new state (so for the first +Concept state, there are 487 instances in the data of a student being correct, certain after hearing a new concept).

Concept Repetition Feature As shown in column 4, the main trend of incorporating concept repetition usually is to give a complex answer question after a concept has been repeated, and especially if the student is correct when addressing a question about the repeated concept. This is intuitive because one would expect that if a concept has been repeated, it signals that the student did not grasp the concept initially and a clarification dialogue was initiated to help the student learn the concept. Once the student answers the repeated concept correctly, it signals that the student understands the concept and that the tutor can once again ask more difficult questions to challenge the student. Given the amount of differences in the new policy and the original policy (10 out of 12 possible), including concept repetition as a state feature has a significant impact on the policy generated.

Frustration Feature Our results show that adding frustration changes the policies the most when the student is frustrated, but when the student isn't frustrated (neutral) the policy stays the same as the baseline with the exception of when the student is Correct and Certain (state 1), and Incorrect and Uncertain (state 6). It should be noted that for states 2 through 6, that frustration occurs very infrequently so the policies generated (CAQ) may not have enough data to be totally reliable. However in state 1, the policy when the student is confident and correct but also frustrated is to simply give a hint or some other form of feedback. In short, adding the frustration feature results in a change in 8 out of 12 policies.

Percent Correctness Feature Finally, the last column, shows the new policy generated for incorporating a simple model of current student performance within the dialog. The main trend is to give a Mix of SAQ and CAQ's. Since the original policy was to give a lot of Mix's in the first place, adding this feature does not result in a large policy change, only 4 differences.

5.2 Feature Comparison

To compare the utility of each of the features, we use three metrics: (1) Diff's (2) % Policy Change, and (3) Expected Cumulative Reward. # of Diff's are the number of states whose policy differs from the baseline policy, The second column of Table 6

#	State	Baseline 2	+Concept	+Frustration	+ % Correctness
1	certain:C	Mix (663)	0: **CAQ** (487) R: **CAQ** (176)	N: **SAQ** (558) F: **NoQ** (105)	H: Mix (650) L: Mix (13)
2	certain:IPC	Mix (251)	0: **SAQ** (190) R: **NoQ** (61)	N: Mix (215) F: **CAQ** (36)	H: Mix (217) L: Mix (34)
3	neutral:C	SAQ (480)	0: **CAQ** (328) R: **CAQ** (152)	N: SAQ (466) F: **CAQ** (14)	H: **Mix** (468) L: **Mix** (12)
4	neutral:IPC	NoQ (377)	0: NoQ (289) R: **Mix** (88)	N: NoQ (364) F: **CAQ** (13)	H: NoQ (320) L: **Mix** (57)
5	uncertain:C	Mix (165)	0: Mix (127) R: **CAQ** (38)	N: Mix (151) F: **CAQ** (14)	H: Mix (156) L: Mix (9)
6	uncertain:IPC	Mix (244)	0: **SAQ** (179) R: **CAQ**(65)	N: **CAQ** (209) F: **CAQ** (35)	H: **CAQ** (182) L: Mix (62)

Table 5: Question Policies

summarizes the amount of Diff's for each new feature compared to Baseline 2. Concept Repetition has the largest number of differences: 10, followed by Frustration, and then Percent Correctness. However, counting the number of differences does not completely describe the effect of the feature on the policy. For example, it is possible that a certain feature may impact the policy for several states that occur infrequently, resulting in a lot of differences but the overall impact may actually be lower than a certain feature that only impacts one state, since that state occurs a majority of the time in the data. So we weight each difference by the number of times that state-action sequence actually occurs in the data and then divide by the total number of state-action sequences. This weighting, % Policy Change (or % P.C.), allows us to more accurately depict the impact of adding the new feature. The third columns shows the weighted figures of % Policy Change. As an additional confirmation of the ranking, we use Expected Cumulative Reward (E.C.R.). One issue with % Policy Change is that it is possible that frequently occurring states have very low V-values so the expected utility from starting the dialogue could potentially be lower than a state feature with low % Policy Change. E.C.R. is calculated by normalizing the V-value of each state by the number of times it occurs as a start state in a dialogue and then summing over all states. The upshot of both metrics is the ranking of the three features remains the same with Concept Repetition effecting the greatest change in what a tutoring system should do; Percent Correctness has the least effect.

We also added a random feature to Baseline 2

State Feature	# Diff's	% P.C.	E.C.R
Concept Repetition	10	80.2%	39.52
Frustration	8	66.4%	31.30
Percent Correctness	4	44.3%	28.17

Table 6: Question Act Results

State Feature	# Diff's	% P.C.	E.C.R
Concept Repetition	4	34.6%	43.43
Frustration	3	6.0%	25.80
Percent Correctness	3	10.3%	26.41

Table 7: Feedback Act Results

with one of two values (0 and 1) to serve as a baseline for the # of Diff's. In a MDP with a large enough corpus to explore, a random variable would not alter the policy, however with a smaller corpus it is possible for such a variable to alter policies. We found that by testing a random feature 40 times and averaging the diffs from each test, resulted in an average diff of 5.1. This means that Percent Correctness effects a smaller amount of change than this random baseline and thus is fairly useless as a feature to add since the random feature is probably capturing some aspect of the data that is more useful. However, the Concept Repetition and Frustration cause more change in the policies than the random feature baseline so one can view them as fairly useful still.

As a final test, we investigated the utility of each feature by using a different tutor action - whether or not the tutor should give simple feedback (SimFeed), or a complex feedback response(ComFeed), or a combination of the two (Mix) (Tetreault and Litman, 2006). The policies and distributions for all features from this previous work are shown in Ta-

#	State	Baseline 2	+Concept	+Frustration	+ % Correctness
1	certain:C	ComFeed (663)	0: ComFeed (487) R: **SimFeed** (176)	N: ComFeed (558) F: **SimFeed** (105)	H: ComFeed (650) L: ComFeed (13)
2	certain:IPC	ComFeed (251)	0: **Mix** (190) R: **Mix** (61)	N: ComFeed (215) F: ComFeed (36)	H: ComFeed (217) L: ComFeed (34)
3	neutral:C	SimFeed (480)	0: **Mix** (328) R: SimFeed (152)	N: SimFeed (466) F: **ComFeed** (14)	H: SimFeed (468) L: **ComFeed** (12)
4	neutral:IPC	Mix (377)	0: Mix (289) R: Mix (88)	N: Mix (364) F: **ComFeed** (13)	H: Mix (320) L: **ComFeed** (57)
5	uncertain:C	ComFeed (165)	0: ComFeed (127) R: ComFeed (38)	N: ComFeed (151) F: ComFeed (14)	H: **Mix** (156) L: ComFeed (9)
6	uncertain:IPC	ComFeed (244)	0: ComFeed (179) R: ComFeed (65)	N: ComFeed (209) F: ComFeed (35)	H: ComFeed (182) L: ComFeed (62)

Table 8: Feedback Policies (summarized from (Tetreault and Litman, 2006))

bles 7 and 8. Basically, we wanted to see if the relative rankings of the three features remained the same for a different action set and whether different action sets evoked different changes in policy. The result is that although the amount of policy change is much lower than when using Questions as the tutor action, the relative ordering of the features is still about the same with Concept Repetition still having the greatest impact on the policy. Interestingly, while Frustration and Percent Correctness have lower diffs, % policy changes, and E.C.R. then their question counterparts (which indicates that those features are less important when considering what type of feedback to give, as opposed to what type of question to give), the E.C.R. for concept repetition with feedback is actually higher than the question case.

6 Related Work

RL has been applied to improve dialogue systems in past work but very few approaches have looked at which features are important to include in the dialogue state. Paek and Chickering's (2005) work on testing the Markov Assumption for Dialogue Systems showed how the state space can be learned from data along with the policy. One result is that a state space can be constrained by only using features that are relevant to receiving a reward. Henderson et al.'s (2005) work focused on learning the best policy by using a combination of reinforcement and supervised learning techniques but also addressed state features by using linear function approximation to deal with large state spaces. Singh et al. (1999) and Frampton et al. (2005) both showed the effect of adding one discourse feature to the student state (dialogue length and user's last dialogue act, respectively) whereas in our work we compare the worth of multiple features. Although Williams et al.'s (2005b) work did not focus on choosing the best state features, they did show that in a noisy environment, Partially-Observable MDP's could be used to build a better model of what state the user is in, over traditional MDP and hand-crafted methods. One major difference between all this related work and ours is that usually the work is focused on how to best deal with ASR errors. Although this is also important in the tutoring domain, our work is novel because it focuses on more semantically-oriented questions.

7 Discussion

In this paper we showed that incorporating more information into a representation of the student state has an impact on what actions the tutor should take. Specifically, we proposed three metrics to determine the relative weight of the three features. Our empirical results indicate that Concept Repetition and Frustration are the most compelling since adding them to the baseline resulted in major policy changes. Percent Correctness had a negligible effect since it resulted in only minute changes to the baseline policy. In addition, we also showed that the relative ranking of these features generalizes across different action sets.

While these features may appear unique to tutoring systems they also have analogs in other dialogue systems as well. Repeating a concept (whether it be a physics term or travel information) is important because it is an implicit signal that there might be some

confusion and a different action is needed when the concept is repeated. Frustration can come from difficulty of questions or from the more frequent problem for any dialogue system, speech recognition errors, so the manner in dealing with it will always be important. Percent Correctness can be viewed as a specific instance of tracking user performance such as if they are continuously answering questions properly or are confused by what the system requests.

With respect to future work, we are annotating more human-computer dialogue data and will triple the size of our test corpus allowing us to create more complicated states since more states will have been explored, and test out more complex tutor actions, such as when to give Hints and Restatements. In the short term, we are investigating whether other metrics such as entropy and confidence bounds can better indicate the usefulness of a feature. Finally, it should be noted that the certainty and frustration feature scores are based on a manual annotation. We are investigating how well an automated certainty and frustration detection algorithm will impact the % Policy Change. Previous work such as (Liscombe et al., 2005) has shown that certainty can be automatically generated with accuracy as high as 79% in comparable human-human dialogues. In our corpus, we achieve an accuracy of 60% in automatically predicting certainty.

8 Acknowledgments

We would like to thank the ITSPOKE and Pitt NLP groups, Pam Jordan, James Henderson, and the three anonymous reviewers for their comments. Support for this research was provided by NSF grants #0325054 and #0328431.

References

K. Bhatt, M. Evens, and S. Argamon. 2004. Hedged responses and expressions of affect in human/human and human computer tutorial interactions. In *Proc. Cognitive Science*.

K. Forbes-Riley and D. Litman. 2005. Using bigrams to identify relationships between student certainness states and tutor responses in a spoken dialogue corpus. In *SIGDial*.

K. Forbes-Riley, D. Litman, A. Huettner, and A. Ward. 2005. Dialogue-learning correlations in spoken dialogue tutoring. In *Artificial Intelligence in Education*.

M. Frampton and O. Lemon. 2005. Reinforcement learning of dialogue strategies using the user's last dialogue act. In *IJCAI Wkshp. on K&R in Practical Dialogue Systems*.

J. Henderson, O. Lemon, and K. Georgila. 2005. Hybrid reinforcement/supervised learning for dialogue policies from communicator data. In *IJCAI Wkshp. on K&R in Practical Dialogue Systems*.

E. Levin and R. Pieraccini. 1997. A stochastic model of computer-human interaction for learning dialogues. In *Proc. of EUROSPEECH '97*.

J. Liscombe, J. Hirschberg, and J. Venditti. 2005. Detecting certainness in spoken tutorial dialogues. In *Interspeech*.

D. Litman and S. Silliman. 2004. Itspoke: An intelligent tutoring spoken dialogue system. In *HLT/NAACL*.

T. Paek and D. Chickering. 2005. The markov assumption in spoken dialogue management. In *6th SIGDial Workshop on Discourse and Dialogue*.

S. Singh, M. Kearns, D. Litman, and M. Walker. 1999. Reinforcement learning for spoken dialogue systems. In *Proc. NIPS '99*.

R. Sutton and A. Barto. 1998. *Reinforcement Learning*. The MIT Press.

J. Tetreault and D. Litman. 2006. Using reinforcement learning to build a better model of dialogue state. In *EACL*.

K. VanLehn, P. Jordan, C. Rosé, D. Bhembe, M. Bottner, A. Gaydos, M Makatchev, U. Pappuswamy, M. Ringenberg, A. Roque, S. Siler, R. Srivastava, and R. Wilson. 2002. The archictecture of why2-atlas: A coach for qualitative physics essay writing. In *Intelligent Tutoring Systems*.

M. Walker. 2000. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *JAIR*, 12.

J. Williams, P. Poupart, and S. Young. 2005a. Factored partially observable markov decision processes for dialogue management. In *IJCAI Wkshp. on K&R in Practical Dialogue Systems*.

J. Williams, P. Poupart, and S. Young. 2005b. Partially obervable markov decision processes with continuous observations for dialogue management. In *SIGDial*.

Backoff Model Training using Partially Observed Data: Application to Dialog Act Tagging

Gang Ji and **Jeff Bilmes**
Department of Electrical Engineering
University of Washington
Seattle, WA 98105-2500
{gang,bilmes}@ee.washington.edu

Abstract

Dialog act (DA) tags are useful for many applications in natural language processing and automatic speech recognition. In this work, we introduce *hidden backoff models* (HBMs) where a large generalized backoff model is trained, using an embedded expectation-maximization (EM) procedure, on data that is partially observed. We use HBMs as word models conditioned on both DAs and (hidden) DA-segments. Experimental results on the ICSI meeting recorder dialog act corpus show that our procedure can strictly increase likelihood on training data and can effectively reduce errors on test data. In the best case, test error can be reduced by 6.1% relative to our baseline, an improvement on previously reported models that also use prosody. We also compare with our own prosody-based model, and show that our HBM is competitive even without the use of prosody. We have not yet succeeded, however, in combining the benefits of both prosody and the HBM.

1 Introduction

Discourse patterns in natural conversations and meetings are well known to provide interesting and useful information about human conversational behavior. They thus attract research from many different and beneficial perspectives. Dialog acts (DAs) (Searle, 1969), which reflect the functions that utterances serve in a discourse, are one type of such patterns. Detecting and understanding dialog act patterns can provide benefit to systems such as automatic speech recognition (ASR) (Stolcke et al., 1998), machine dialog translation (Lee et al., 1998), and general natural language processing (NLP) (Jurafsky et al., 1997b; He and Young, 2003). DA pattern recognition is an instance of "tagging." Many different techniques have been quite successful in this endeavor, including hidden Markov models (Jurafsky et al., 1997a; Stolcke et al., 1998), semantic classification trees and polygrams (Mast et al., 1996), maximum entropy models (Ang et al., 2005), and other language models (Reithinger et al., 1996; Reithinger and Klesen, 1997). Like other tagging tasks, DA recognition can also be achieved using conditional random fields (Lafferty et al., 2001; Sutton et al., 2004) and general discriminative modeling on structured outputs (Bartlett et al., 2004). In many sequential data analysis tasks (speech, language, or DNA sequence analysis), standard dynamic Bayesian networks (DBNs) (Murphy, 2002) have shown great flexibility and are widely used. In (Ji and Bilmes, 2005), for example, an analysis of DA tagging using DBNs is performed, where the models avoid label bias by structural changes and avoid data sparseness by using a generalized backoff procedures (Bilmes and Kirchhoff, 2003).

Most DA classification procedures assume that within a sentence of a particular fixed DA type, there is a fixed word distribution over the entire sentence. Similar to (Ma et al., 2000) (and see citations therein), we have found, however, that intra-

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 280–287,
New York, June 2006.

sentence discourse patterns are inherently dynamic. Moreover, the patterns are specific to each type of DA, meaning a sentence will go through a DA-specific sequence of sub-DA phases or "states." A generative description of this phenomena is that a DA is first chosen, and then words are generated according to both the DA and to the relative position of the word in that sentence. For example, a "statement" (one type of DA) can consist of a subject (noun phrase), verb phrase, and object (noun phrase). This particular sequence might be different for a different DA (e.g., a "back-channel"). Our belief is that explicitly modeling these internal states can help a DA-classification system in conversational meetings or dialogs.

In this work, we describe an approach that is motivated by several aspects of the typical DA-classification procedure. First, it is rare to have sub-DAs labeled in training data, and indeed this is true of the corpus (Shriberg et al., 2004) that we use. Therefore, some form of unsupervised clustering or pre-shallow-parsing of sub-DAs must be performed. In such a model, these sub-DAs are essentially unknown hidden variables that ideally could be trained with an expectation-maximization (EM) procedure. Second, when training models of language, it is necessary to employ some form of smoothing methodology since otherwise data-sparseness would render standard maximum-likelihood trained models useless. Third, discrete conditional probability distributions formed using backoff models that have been smoothed (particularly using modified Kneser-Ney (Chen and Goodman, 1998)) have been extremely successful in many language modeling tasks. Training backoff models, however, requires that all data is observed so that data counts can be formed. Indeed, our DA-specific word models (implemented via backoff) will also need to condition on the current sub-DA, which at training time is unknown. We therefore have developed a procedure that allows us to train generalized backoff models (Bilmes and Kirchhoff, 2003), even when some or all of the variables involved in the model are *hidden*. We thus call our models *hidden backoff model*s (HBMs). Our method is indeed a form of embedded EM training (Morgan and Bourlard, 1990), and more generally is a specific form of EM (Neal and Hinton, 1998). Our approach is similar to (Ma et al., 2000), except our underlying language models are backoff-based and thus retain the benefits of advanced smoothing methods, and we utilize both a normal and a backoff EM step as will be seen. We moreover wrap up the above ideas in the framework of dynamic Bayesian networks, which are used to represent and train all of our models.

We evaluate our methods on the ICSI meeting recorder dialog act (MRDA) (Shriberg et al., 2004) corpus, and find that our novel hidden backoff model can significantly improve dialog tagging accuracy. With a different number of hidden states for each DA, a relative reduction in tagging error rate as much as 6.1% can be achieved. Our best HBM result shows an accuracy that improves on the best known (to our knowledge) result on this corpora which is one that uses acoustic prosody as a feature. We have moreover developed our own prosody model and while we have not been able to usefully employ both prosody and the HBM technique together, our HBM is competitive in this case as well. Furthermore, our results show the effectiveness of our embedded EM procedure, as we demonstrate that it increases training log likelihoods, while simultaneously reducing error rate.

Section 2 briefly summarizes our baseline DBN-based models for DA tagging tasks. In Section 3, we introduce our HBMs. Section 4 contains experimental evaluations on the MRDA corpus and finally Section 5 concludes.

2 DBN-based Models for Tagging

Dynamic Bayesian networks (DBNs) (Murphy, 2002) are widely used in sequential data analysis such as automatic speech recognition (ASR) and DNA sequencing analysis (Durbin et al., 1999). A hidden Markov model (HMM) for DA tagging as in (Stolcke et al., 1998) is one such instance.

Figure 1 shows a generative DBN model that will be taken as our baseline. This DBN shows a prologue (the first time slice of the model), an epilogue (the last slice), and a chunk that is repeated sufficiently to fit the entire data stream. In this case, the data stream consists of the words of a meeting conversation, where individuals within the meeting (hopefully) take turns speaking. In our model, the entire meeting conversation, and all turns of all

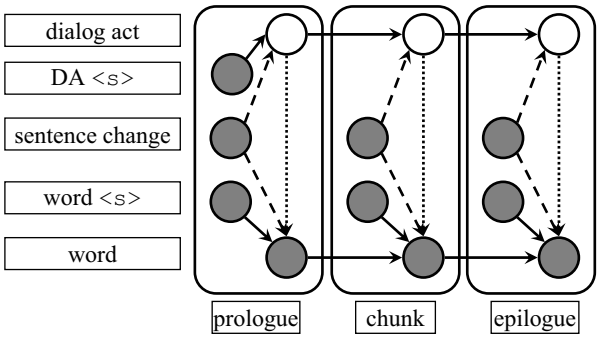


Figure 1: Baseline generative DBN for DA tagging.

speakers, are strung together into a single stream rather than treating each turn in the meeting individually. This approach has the benefit that we are able to integrate a temporal DA-to-DA model (such as a DA bigram).

In all our models, to simplify we assume that the sentence change information is known (as is common with this corpus (Shriberg et al., 2004)). We next describe Figure 1 in detail. Normally, the *sentence change* variable is not set, so that we are within a sentence (or a particular DA). When a sentence change does not occur, the DA stays the same from slice to slice. During this time, we use a DA-specific language model (implemented via a backoff strategy) to score the words within the current DA.

When a sentence change event does occur, a new DA is predicted based on the DA from the previous sentence (using a DA bigram). At the beginning of a sentence, rather than conditioning on the last word of the previous sentence, we condition on the special start of sentence <s> token, as shown in the figure by having a special parent that is used only when *sentence change* is true. Lastly, at the very beginning of a meeting, a special start of DA token is used.

The joint probability under this baseline model is written as follows:

$$P(W, D) = \prod_k P(d_k|d_{k-1}) \cdot \prod_i P(w_{k,i}|w_{k,i-1}, d_k), \quad (1)$$

where $W = \{w_{k,i}\}$ is the word sequence, $D = \{d_k\}$ is the DA sequence, d_k is the DA of the k-th sentence, and $w_{k,i}$ is the i-th word of the k-th sentence in the meeting.

Because all variables are observed when training our baseline, we use the SRILM toolkit (Stolcke, 2002), modified Kneser-Ney smoothing (Chen and Goodman, 1998), and factored extensions (Bilmes and Kirchhoff, 2003). In evaluations, the Viterbi algorithm (Viterbi, 1967) can be used to find the best DA sequence path from the words of the meeting according to the joint distribution in Equation (1).

3 Hidden Backoff Models

When analyzing discourse patterns, it can be seen that sentences with different DAs usually have different internal structures. Accordingly, in this work we do not assume sentences for each dialog act have the same hidden state patterns. For instance (and as mentioned above), a statement can consist of a noun followed by a verb phase.

A problem, however, is that sub-DAs are not annotated in our training corpus. While clustering and annotation of these phrases is already a widely developed research topic (Pieraccini and Levin, 1991; Lee et al., 1997; Gildea and Jurafsky, 2002), in our approach we use an EM algorithm to learn these hidden sub-DAs in a data-driven fashion. Pictorially, we add a layer of hidden states to our baseline DBN as illustrated in Figure 2.

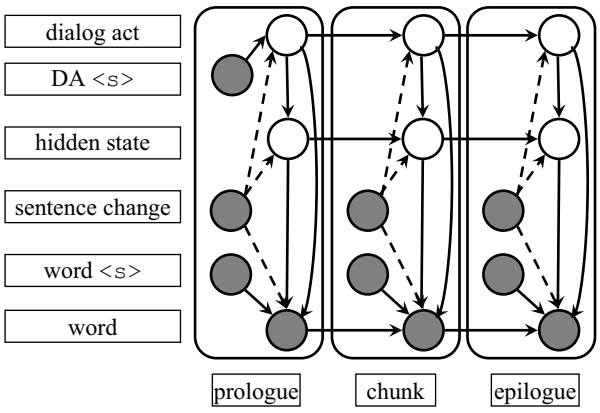


Figure 2: Hidden backoff model for DA tagging.

Under this model, the joint probability is:

$$P(W, S, D) = \prod_k P(d_k|d_{k-1}) \cdot \prod_i [P(s_{k,i}|s_{k,i-1}, d_k) \cdot P(w_{k,i}|w_{k,i-1}, s_{k,i}, d_k)], \quad (2)$$

where $S = \{s_{k,i}\}$ is the hidden state sequence, $s_{k,i}$ is the hidden state at the i-th position of the k-th sentence, and other variables are the same as before.

Similar to our baseline model, the DA bigram $P(d_k|d_{k-1})$ can be modeled using a backoff bigram. Moreover, if the states $\{s_{k,i}\}$ are known during training, the word prediction probability $P(w_{k,i}|w_{k,i-1}, s_{k,i}, d_k)$ can also use backoff and be trained accordingly. The hidden state sequence is unknown, however, and thus cannot be used to produce a standard backoff model. What we desire is an ability to utilize a backoff model (to mitigate data sparseness effects) while simultaneously retaining the state as a hidden (rather than an observed) variable, and also have a procedure that trains the entire model to improve overall model likelihood.

Expectation-maximization (EM) algorithms are well-known to be able to train models with hidden states. Furthermore, standard advanced smoothing methods such as modified Kneser-Ney smoothing (Chen and Goodman, 1998) utilize integer counts (rather than fractional ones), and they moreover need "meta" counts (or counts of counts). Therefore, in order to train this model, we propose an embedded training algorithm that cycles between a standard EM training procedure (to train the hidden state distribution), and a stage where the most likely hidden states (and their counts and meta counts) are used externally to train a backoff model. This procedure can be described in detail as follows:

Input : W — meeting word sequence
Input : D — DA sequence
Output : $P(s_{k,i}|s_{k,i-1})$ - state transition CPT
Output : $P(w_{k,i}|w_{k,i-1}, s_{k,i}, d_k)$ - word model

randomly generate a sequence S;
backoff train $P(w_{k,i}|w_{k,i-1}, s_{k,i}, d_k)$;
while *not "converged"* **do**
  EM train $P(s_{k,i}|s_{k,i-1})$;
  calculate best $\bar{S}$ sequence by Viterbi;
  backoff train $P(w_{k,i}|w_{k,i-1}, \bar{s}_{k,i}, d_k)$;
end

Algorithm 1: Embedded training for HBMs

In the algorithm, the input contains words and a DA for each sentence in the meeting. The output is the corresponding conditional probability table (CPT) for hidden state transitions, and a backoff model for word prediction. Because we train the backoff model when some of the variables are hidden, we call the result a *hidden backoff model*. While we have seen embedded Viterbi training used in the past for simultaneously training heterogeneous models (e.g., Markov chains and Neural Networks (Morgan and Bourlard, 1990)), this is the first instance of training backoff-models that involve hidden variables that we are aware of.

While embedded Viterbi estimation is not guaranteed to have the same convergence (or fixed-point under convergence) as normal EM (Lember and Koloydenko, 2004), we find empirically this to be the case (see examples below). Moreover, our algorithm can easily be modified so that instead of taking a Viterbi alignment in step 5, we instead use a set of random samples generated under the current model. In this case, it can be shown using a law-of-large numbers argument that having sufficient samples guarantees the algorithm will converge (we will investigate this modification in future work).

Of course, when decoding with such a model, a conventional Viterbi algorithm can still be used to calculate the best DA sequence.

4 Experimental Results

We evaluated our hidden backoff model on the ICSI meeting recorder dialog act (MRDA) corpus (Shriberg et al., 2004). MRDA is a rich data set that contains 75 natural meetings on different topics with each meeting involving about 6 participants. DA annotations from ICSI were based on a previous approach in (Jurafsky et al., 1997b) with some adaptation for meetings in a number of ways described in (Bhagat et al., 2003). Each DA contains a main tag, several optional special tags and an optional "disruption" form. The total number of distinct DAs in the corpus is as large as 1260. In order to make the problem comparable to other work (Ang et al., 2005), a DA tag sub-set is used in our experiments that contains back channels (b), place holders (h), questions (q), statements (s), and disruptions (x). In our evaluations, among the entire 75 conversations, 51 are used as the training set, 11 are used as the development set, 11 are used as test set, and the remaining 3 are not used. For each experiment, we used a genetic algorithm to search for the best factored language model structure on the development set and

we report the best results.

Our baseline system is the generative model shown in Figure 1 and uses a backoff implementation of the word model, and is optimized on the development set. We use the SRILM toolkit with extensions (Bilmes and Kirchhoff, 2003) to train, and use GMTK (Bilmes and Zweig, 2002) for decoding. Our baseline system has an error rate of 19.7% on the test set, which is comparable to other approaches on the same task (Ang et al., 2005).

4.1 Same number of states for all DAs

To compare against our baseline, we use HBMs in the model shown in Figure 2. To train, we followed Algorithm 1 as described before and as is here detailed in Figure 3.

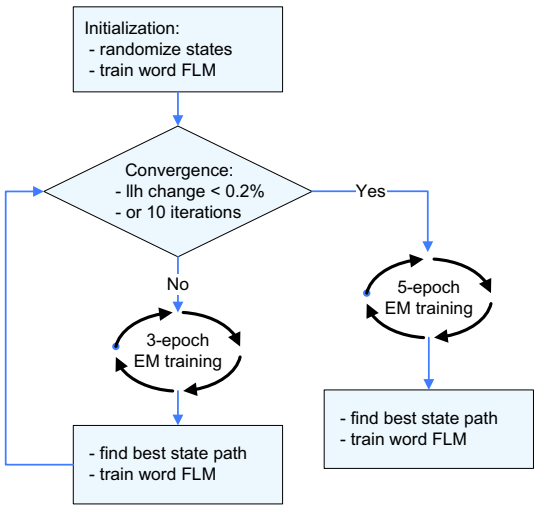


Figure 3: Embedded training: llh = log likelihood

In this implementation, an upper triangular matrix (with self-transitions along the diagonal) is used for the hidden state transition probability table so that sub-DA states only propagate in one direction. When initializing the hidden state sequence of a DA, we expanded the states uniformly along the sentence. This initial alignment is then used for HBM training. In the word models used in our experiments, the backoff path first drops previous words, then does a parallel backoff to hidden state and DA using a mean combination strategy.

The HBM thus obtained was then fed into the main loop of our embedded EM algorithm. The training was considered to have "converged" if either it exceeded 10 iterations (which never happened) or the relative log likelihood change was less than 0.2%. Within each embedded iteration, three EM epochs were used. After each EM iteration, a Viterbi alignment was performed thus obtaining what we expect to be a better hidden state alignment. This updated alignment, was then used to train a new HBM. The newly generated model was then fed back into the embedded training loop until it converged. After the procedure met our convergence criteria, an additional five EM epochs were carried out in order to provide a good hidden state transition probability table. Finally, after Viterbi alignment and text generation was performed, the word HBM was trained from the best state sequence.

To evaluate our hidden backoff model, the Viterbi algorithm was used to find the best DA sequence according to test data, and the tagging error rates were calculated. In our first experiment, an equal number of hidden states for all DAs were used in each model. The effect of this number on the accuracy of DA tagging is shown in Table 1.

Table 1: HBMs, different numbers of hidden states.

# states	error	improvement
baseline	19.7%	–
2-state	18.7%	5.1%
3-state	19.5%	1.0%

For the baseline system, the backoff path first drops `dialog act`, and for the HBMs, all backoff paths drop `hidden state` first and drop `DA` second. From Table 1 we see that with two hidden states for every DA the system can reduce the tagging error rate by more than 5% relative. As a comparison, in (Ang et al., 2005), where conditional maximum entropy models (which are conditionally trained) are used, the error rate is 18.8% when using both word and acoustic prosody features, and and 20.5% without prosody. When the number of hidden states increases to 3, the improvement decreases even though it is still (very slightly) better than the baseline. We believe the reasons are as follows: First, assuming different DAs have the same number of hidden states may not be appropriate. For example, back channels usually have shorter sentences and are constant in discourse pattern over a DA. On the other hand,

questions and statements typically have longer, and more complex, discourse structures. Second, even under the same DA, the structure and inherent length of sentence can vary. For example, "yes" can also be a statement even though it has only one word. Therefore, one-word statements need completely different hidden state patterns than those in subject-verb-object like statements — having one monolithic 3-state model for statements might be inappropriate. This issue is discussed further in Section 4.4.

4.2 Different states for different DAs

In order to mitigate the first problem described above, we allow different numbers of hidden states for each DA. This, however, leads to a combinatorial explosion of possibilities if done in a naïve fashion. Therefore, we attempted only a small number of combinations based on the statistics of numbers of words in each DA given in Table 2.

Table 2: Length statistics of different DAs.

DA	mean	median	std	p
(b)	1.0423	1	0.2361	0.4954
(h)	1.3145	1	0.7759	0.4660
(q)	6.5032	5	6.3323	0.3377
(s)	8.6011	7	7.8380	0.3013
(x)	1.7201	1	1.1308	0.4257

Table 2 shows the mean and median number of words per sentence for each DA as well as the standard deviation. Also, the last column provides the p value according to fitting the length histogram to a geometric distribution $(1-p)^n p$. As we expected, back channels (b) and place holders (h) tend to have shorter sentences while questions (q) and statements (s) have longer ones. From this analysis, we use fewer states for (b) and (h) and more states for (q) and (s). For disruptions (x), the standard deviation of number of words histogram is relatively high compared with (b) and (h), so we also used more hidden states in this case. In our experimental results below, we used one state for (b) and (h), and various numbers of hidden states for other DAs. Tagging error rates are shown in Table 3.

From Table 3, we see that using different numbers of hidden states for different DAs can produce better models. Among all the experiments we performed, the best case is given by three states for (q), two states for (s) and (x), and one state for (b) and (h). This combination gives 6.1% relative reduction of error rate from the baseline.

Table 3: Number of hidden states for different DAs.

b	h	q	s	x	error	improvement
1	1	4	4	1	18.9%	4.1%
1	1	3	3	2	18.9%	4.1%
1	1	2	2	2	18.7%	5.1%
1	1	3	2	2	18.6%	5.6%
1	1	3	2	2	18.5%	6.1%

4.3 Effect of embedded EM training

Incorporating backoff smoothing procedures into Bayesian networks (and hidden variable training in particular) can show benefits for any data domain where smoothing is necessary. To understand the properties of our algorithm a bit better, after each training iteration using a partially trained model, we calculated both the log likelihood of the training set and the tagging error rate of the test data. Figure 4 shows these results using the best configuration from the previous section (three states for (q), two for (s)/(x) and one for (b)/(h)). This example is typical of the convergence we see of Algorithm 1, which empirically suggests that our procedure may be similar to a generalized EM (Neal and Hinton, 1998).

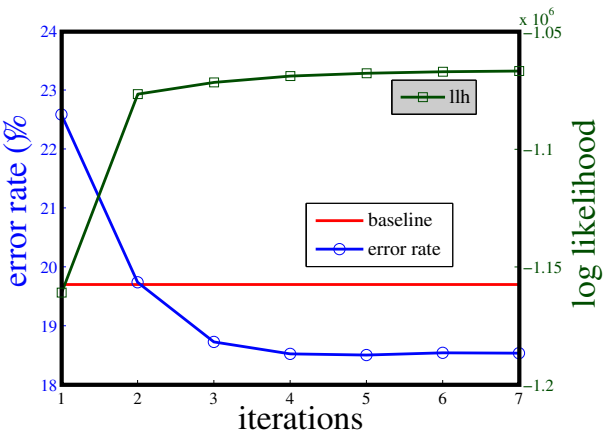


Figure 4: Embedded EM training performance.

We find that the log likelihood after each EM training is strictly increasing, suggesting that our embedded EM algorithm for hidden backoff models

is improving the overall joint likelihood of the training data according to the model. This strict increase of likelihood combined with the fact that Viterbi training does not have the same theoretical convergence guarantees as does normal EM indicates that more detailed theoretical analysis of this algorithm used with these particular models is desirable.

From the figure we also see that both the log likelihood and tagging error rate "converge" after around four iterations of embedded training. This quick convergence indicates that our embedded training procedure is effective. The leveling of the error rates after several iterations shows that model over-fitting appears not to be an issue presumably due to the smoothed embedded backoff models.

4.4 Discussion and Error Analysis

A large portion of our tagging errors are due to confusing the DA of short sentences such as "yeah", and "right". The sentence, "yeah" can either be a back channel or an affirmative statement. There are also cases where "yeah?" is a question. These types of confusions are difficult to remove in the prosody-less framework but there are several possibilities. First, we can allow the use of a "fork and join" transition matrix, where we fork to each DA-specific condition (e.g., short or long) and join thereafter. Alternatively, hidden Markov chain structuring algorithms or context (i.e., conditioning the number of sub-DAs on the previous DA) might be helpful.

Finding a proper number of hidden states for each DA is also challenging. In our preliminary work, we simply explored different combinations using simple statistics of the data. A systematic procedure would be more beneficial. In this work, we also did not perform any hidden state tying within different DAs. In practice, some states in statements should be able to be beneficially tied with other states within questions. Our results show that having three states for all DAs is not as good as two states for all. But with tying, more states might be more successfully used.

4.5 Influence of Prosody Cues

It has been shown that prosody cues provide useful information in DA tagging tasks (Shriberg et al., 1998; Ang et al., 2005). We also incorporated prosody features in our models. We used ESPS `get_f0` based on RAPT algorithm (Talkin, 1995) to get F_0 values. For each speaker, mean and variance normalization is performed. For each word, a linear regression is carried on the normalized F_0 values. We quantize the slope values into 20 bins and treat those as prosody features associated with each word. After adding the prosody features, the simple generative model as shown in Figure 5 gives 18.4% error rate, which is 6.6% improvement over our baseline. There is no statistical difference between the best performance of this prosody model and the earlier best HBM. This implies that the HBM can obtain as good performance as a prosody-based model but without using prosody.

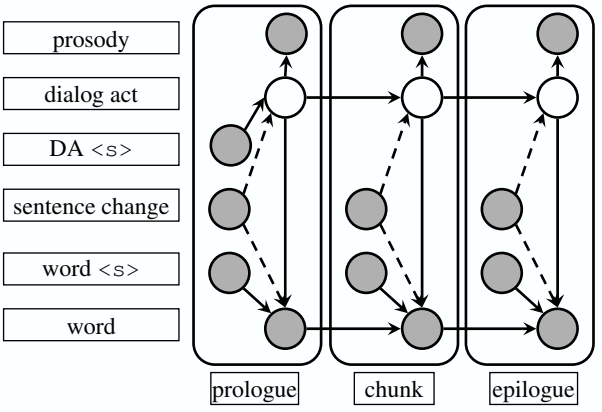


Figure 5: Generative prosody model for DA tagging.

The next obvious step is to combine an HBM with the prosody information. Strangely, even after experimenting with many different models (including ones where prosody depends on DA; prosody depends on DA and the hidden state; prosody depends on DA, hidden state, and word; and many variations thereof), we were unsuccessful in obtaining a complementary benefit when using both prosody and an HBM. One hypothesis is that our prosody features are at the word-level (rather than at the DA level). Another problem might be the small size of the MRDA corpus relative to the model complexity. Yet a third hypothesis is that the errors corrected by both methods are the same — indeed, we have verified that the corrected errors overlap by more than 50%. We plan further investigations in future work.

5 Conclusions

In this work, we introduced a training method for *hidden backoff models* (HBMs) to solve a problem in DA tagging where smoothed backoff models involving training-time hidden variables are useful. We tested this procedure in the context of dynamic Bayesian networks. Different hidden states were used to model different positions in a DA. According to empirical evaluations, our embedded EM algorithm effectively increases log likelihood on training data and reduces DA tagging error rate on test data. If different numbers of hidden states are used for different DAs, we find that our prosody-independent HBM reduces the tagging error rate by 6.1% relative to the baseline, a result that improves upon previously reported work that uses prosody, and that is comparable to our own new result that also incorporates prosody. We have not yet been able to combine the benefits of both an HBM and prosody information. This material is based upon work supported by the National Science Foundation under Grant No. IIS-0121396.

References

J. Ang et al. 2005. Automatic dialog act segmentation and classification in multiparty meetings. In *ICASSP*.

P. Bartlett et al. 2004. Exponentiated gradient algorithms for large-margin structured classification. In *NIPS*.

S. Bhagat et al. 2003. Labeling guide for dialog act tags in the meeting recordering meetings. Technical Report 2, International Computer Science Insititute.

J. Bilmes and K. Kirchhoff. 2003. Factored language models and generalized parallel backoff. In *Human Lang. Tech., North American Chapter of Asssoc. Comp. Ling.*, Edmonton, Alberta, May/June.

J. Bilmes and G. Zweig. 2002. The Graphical Models Toolkit: An open source software system for speech and time-series processing. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*.

S. Chen and J. Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical report, Computer Science Group, Harvard University.

R. Durbin et al. 1999. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press.

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

Y. He and S. Young. 2003. A data-driven spoken language understanding system. In *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 583–588.

G. Ji and J. Bilmes. 2005. Dialog act tagging using graphical models. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Philadelphia, PA, March.

D. Jurafsky et al. 1997a. Automatic detection of discourse structure for speech recognition and understanding. In *Proc. IEEE Workshop on Speech Recognition and Understanding*.

D. Jurafsky et al. 1997b. Switchboard SWBD-DAMSL shallow-discourse-function annotation coders manual. Technical Report 97-02, Institute of Cognitive Science, University of Colorado.

J. Lafferty et al. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

K. Lee et al. 1997. Restricted representation of phrase structure grammar for building a tree annotated corpus of Korean. *Natural Language Engineering*, 3(2-3):215–230.

H. Lee et al. 1998. Speech act analysis model of Korean utterances for automatic dialog translation. *J. KISS(B) (Software and Applications)*, 25(10):1443–1452.

J. Lember and A. Koloydenko. 2004. Adjusted viterbi training. a proof of concept. *In Submission*.

K. Ma et al. 2000. Bi-modal sentence structure for language modeling. *Speech Communication*, 31(1):51–67.

M. Mast et al. 1996. Automatic classification of dialog acts with semantic classification trees and polygrams. *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, pages 217–229.

N. Morgan and H. Bourlard. 1990. Continuous speech recognition using multilayer perceptrons with hidden Markov models. In *ICASSP*, pages 413–416.

K. Murphy. 2002. *Dynamic Bayesian Networks, Representation, Inference, and Learning*. Ph.D. thesis, MIT, Dept. Computer Science.

R. Neal and G. Hinton. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368. Dordrecht: Kluwer Academic Publishers.

R. Pieraccini and E. Levin. 1991. Stochastic representation of semantic structure for speech understanding. In *Eurospeech*, volume 2, pages 383–386.

N. Reithinger and M. Klesen. 1997. Dialogue act classification using language models. In *Eurospeech*.

N. Reithinger et al. 1996. Predicting dialogue acts for a speech-to-speech translation system. In *ICLSP*, pages 654–657.

J. Searle. 1969. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press.

E. Shriberg et al. 1998. Can prosody aid the automatic classification of dialog acts in conversational speech? *Language and Speech*, 41(3–4):439–487.

E. Shriberg et al. 2004. The ICSI meeting recorder dialog act (MRDA) corpus. In *Proc. of the 5th SIGdial Workshop on Discourse and Dialogue*, pages 97–100.

A. Stolcke et al. 1998. Dialog act modeling for conversational speech. In *Proc. AAAI Spring Symp. on Appl. Machine Learning to Discourse Processing*, pages 98–105.

A. Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *ICLSP*, volume 2, pages 901–904.

C. Sutton et al. 2004. Dynamic conditional random fields: factorized probabilistic models for labeling and segmenting sequence data. In *ICML*.

D. Talkin. 1995. A robust algorithm for pitch tracking (rapt). In W. B. Kleijn and K.K. Paliwal, editors, *Speech Coding and Synthesis*, pages 495–518. Elsevier Science.

A. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. on Information Theory*, 13(2):260–269.

Exploring Syntactic Features for Relation Extraction using a Convolution Tree Kernel

Min ZHANG **Jie ZHANG** **Jian SU**

Institute for Infocomm Research

21 Heng Mui Keng Terrace, Singapore 119613

{mzhang, zhangjie, sujian}@i2r.a-star.edu.sg

Abstract

This paper proposes to use a convolution kernel over parse trees to model syntactic structure information for relation extraction. Our study reveals that the syntactic structure features embedded in a parse tree are very effective for relation extraction and these features can be well captured by the convolution tree kernel. Evaluation on the ACE 2003 corpus shows that the convolution kernel over parse trees can achieve comparable performance with the previous best-reported feature-based methods on the 24 ACE relation subtypes. It also shows that our method significantly outperforms the previous two dependency tree kernels on the 5 ACE relation major types.

1 Introduction

Relation extraction is a subtask of information extraction that finds various predefined semantic relations, such as location, affiliation, rival, etc., between pairs of entities in text. For example, the sentence "George Bush is the president of the United States." conveys the semantic relation "President" between the entities "George Bush" (PER) and "the United States" (GPE: a Geo-Political Entity --- an entity with land and a government (ACE, 2004)).

Prior feature-based methods for this task (Kambhatla 2004; Zhou et al., 2005) employed a large amount of diverse linguistic features, varying from lexical knowledge, entity mention information to syntactic parse trees, dependency trees and semantic features. Since a parse tree contains rich syntactic structure information, in principle, the features extracted from a parse tree should contribute much more to performance improvement for relation extraction. However it is reported (Zhou et al., 2005; Kambhatla, 2004) that hierarchical structured syntactic features contributes less to performance improvement. This may be mainly due to the fact that the syntactic structure information in a parse tree is hard to explicitly describe by a vector of linear features. As an alternative, kernel methods (Collins and Duffy, 2001) provide an elegant solution to implicitly explore tree structure features by directly computing the similarity between two trees. But to our surprise, the sole two-reported dependency tree kernels for relation extraction on the ACE corpus (Bunescu and Mooney, 2005; Culotta and Sorensen, 2004) showed much lower performance than the feature-based methods. One may ask: are the syntactic tree features very useful for relation extraction? Can tree kernel methods effectively capture the syntactic tree features and other various features that have been proven useful in the feature-based methods?

In this paper, we demonstrate the effectiveness of the syntactic tree features for relation extraction and study how to capture such features via a convolution tree kernel. We also study how to select the optimal feature space (e.g. the set of sub-trees to represent relation instances) to optimize the system performance. The experimental results show that the convolution tree kernel plus entity features achieves slightly better performance than the previous best-reported feature-based methods. It also shows that our method significantly outperforms the two dependency tree kernels (Bunescu and Mooney, 2005; Culotta and Sorensen, 2004) on the 5 ACE relation types.

The rest of the paper is organized as follows. In Section 2, we review the previous work. Section 3 discusses our tree kernel based learning algorithm.

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 288–295,
New York, June 2006.

Section 4 shows the experimental results and compares our work with the related work. We conclude our work in Section 5.

2 Related Work

The task of relation extraction was introduced as a part of the Template Element task in MUC6 and formulated as the Template Relation task in MUC7 (MUC, 1987-1998).

Miller et al. (2000) address the task of relation extraction from the statistical parsing viewpoint. They integrate various tasks such as POS tagging, NE tagging, template extraction and relation extraction into a generative model. Their results essentially depend on the entire full parse tree.

Kambhatla (2004) employs Maximum Entropy models to combine diverse lexical, syntactic and semantic features derived from the text for relation extraction. Zhou et al. (2005) explore various features in relation extraction using SVM. They conduct exhaustive experiments to investigate the incorporation and the individual contribution of diverse features. They report that chunking information contributes to most of the performance improvement from the syntactic aspect.

The features used in Kambhatla (2004) and Zhou et al. (2005) have to be selected and carefully calibrated manually. Kambhatla (2004) use the path of non-terminals connecting two mentions in a parse tree as the parse tree features. Besides, Zhou et al. (2005) introduce additional chunking features to enhance the parse tree features. However, the hierarchical structured information in the parse trees is not well preserved in their parse tree-related features.

As an alternative to the feature-based methods, kernel methods (Haussler, 1999) have been proposed to implicitly explore features in a high dimensional space by employing a kernel function to calculate the similarity between two objects directly. In particular, the kernel methods could be very effective at reducing the burden of feature engineering for structured objects in NLP research (Culotta and Sorensen, 2004). This is because a kernel can measure the similarity between two discrete structured objects directly using the original representation of the objects instead of explicitly enumerating their features.

Zelenko et al. (2003) develop a tree kernel for relation extraction. Their tree kernel is recursively defined in a top-down manner, matching nodes from roots to leaf nodes. For each pair of matching nodes, a subsequence kernel on their child nodes is invoked, which matches either contiguous or sparse subsequences of node. Culotta and Sorensen (2004) generalize this kernel to estimate similarity between dependency trees. One may note that their tree kernel requires the matchable nodes must be at the same depth counting from the root node. This is a strong constraint on the matching of syntax so it is not surprising that the model has good precision but very low recall on the ACE corpus (Zhao and Grishman, 2005). In addition, according to the top-down node matching mechanism of the kernel, once a node is not matchable with any node in the same layer in another tree, all the sub-trees below this node are discarded even if some of them are matchable to their counterparts in another tree.

Bunescu and Mooney (2005) propose a shortest path dependency kernel for relation extraction. They argue that the information to model a relationship between entities is typically captured by the shortest path between the two entities in the dependency graph. Their kernel is very straightforward. It just sums up the number of common word classes at each position in the two paths. We notice that one issue of this kernel is that they limit the two paths must have the same length, otherwise the kernel similarity score is zero. Therefore, although this kernel shows non-trivial performance improvement than that of Culotta and Sorensen (2004), the constraint makes the two dependency kernels share the similar behavior: good precision but much lower recall on the ACE corpus.

Zhao and Grishman (2005) define a feature-based composite kernel to integrate diverse features. Their kernel displays very good performance on the 2004 version of ACE corpus. Since this is a feature-based kernel, all the features used in the kernel have to be explicitly enumerated. Similar with the feature-based method, they also represent the tree feature as a link path between two entities. Therefore, we wonder whether their performance improvement is mainly due to the explicitly incorporation of diverse linguistic features instead of the kernel method itself.

The above discussion suggests that the syntactic features in a parse tree may not be fully utilized in the previous work, whether feature-based or kernel-based. We believe that the syntactic tree features could play a more important role than that

reported in the previous work. Since convolution kernels aim to capture structural information in terms of sub-structures, which providing a viable alternative to flat features, in this paper, we propose to use a convolution tree kernel to explore syntactic features for relation extraction. To our knowledge, convolution kernels have not been explored for relation extraction[1].

3 Tree Kernels for Relation Extraction

In this section, we discuss the convolution tree kernel associated with different relation feature spaces. In Subsection 3.1, we define seven different relation feature spaces over parse trees. In Subsection 3.2, we introduce a convolution tree kernel for relation extraction. Finally we compare our method with the previous work in Subsection 3.3.

3.1 Relation Feature Spaces

In order to study which relation feature spaces (i.e., which portion of parse trees) are optimal for relation extraction, we define seven different relation feature spaces as follows (as shown in Figure 1):

(1) Minimum Complete Tree (MCT):

It is the complete sub-tree rooted by the node of the nearest common ancestor of the two entities under consideration.

(2) Path-enclosed Tree (PT):

It is the smallest common sub-tree including the two entities. In other words, the sub-tree is enclosed by the shortest path linking the two entities in the parse tree (this path is also typically used as the path tree features in the feature-based methods).

(3) Chunking Tree (CT):

It is the base phrase list extracted from the **PT**. We prune out all the internal structures of the **PT** and only keep the root node and the base phrase list for generating the chunking tree.

(4) Context-Sensitive Path Tree (CPT):

It is the **PT** extending with the 1st left sibling of the node of entity 1 and the 1st right sibling of the node of entity 2. If the sibling is unavailable, then we move to the parent of current node and repeat the same process until the sibling is available or the root is reached.

(5) Context-Sensitive Chunking Tree (CCT):

It is the **CT** extending with the 1st left sibling of the node of entity 1 and the 1st right sibling of the node of entity 2. If the sibling is unavailable, the same process as generating the **CPT** is applied. Then we do a further pruning process to guarantee that the context structures of the **CCT** is still a list of base phrases.

(6) Flattened PT (FPT):

We define two criteria to flatten the **PT** in order to generate the **Flattened Parse tree**: if the in and out arcs of a non-terminal node (except POS node) are both single, the node is to be removed; if a node has the same phrase type with its father node, the node is also to be removed.

(7) Flattened CPT (FCPT):

We use the above two criteria to flatten the **CPT** tree to generate the **Flattened CPT**.

Figure 1 in the next page illustrates the different sub-tree structures for a relation instance in sentence "*Akyetsu testified he was powerless to stop the merger of an estimated 2000 ethnic **Tutsi's** in the **district** of Tawba.*". The relation instance is an example excerpted from the ACE corpus, where an ACE-defined relation "AT.LOCATED" exists between the entities "***Tutsi's***" (PER) and "***district***" (GPE).

We use Charniak's parser (Charniak, 2001) to parse the example sentence. Due to space limitation, we do not show the whole parse tree of the entire sentence here. Tree T_1 in Figure 1 is the **MCT** of the relation instance example, where the sub-structure circled by a dashed line is the **PT**. For clarity, we re-draw the **PT** as in T_2. The only difference between the **MCT** and the **PT** lies in that the **MCT** does not allow the partial production rules. For instance, the most-left two-layer sub-tree [NP [DT … E1-O-PER]] in T_1 is broken apart in T_2. By comparing the performance of T_1 and T_2, we can test whether the sub-structures with partial production rules as in T_2 will decrease performance. T_3 is the **CT**. By comparing the performance of T_2 and T_3, we want to study whether the chunking information or the parse tree is more effective

[1] Convolution kernels were proposed as a concept of kernels for a discrete structure by Haussler (1999) in machine learning study. This framework defines a kernel between input objects by applying convolution "sub-kernels" that are the kernels for the decompositions (parts) of the objects. Convolution kernels are abstract concepts, and the instances of them are determined by the definition of "sub-kernels". The *Tree Kernel* (Collins and Duffy, 2001), *String Subsequence Kernel* (SSK) (Lodhi et al., 2002) and *Graph Kernel* (HDAG Kernel) (Suzuki et al., 2003) are examples of convolution kernels instances in the NLP field.

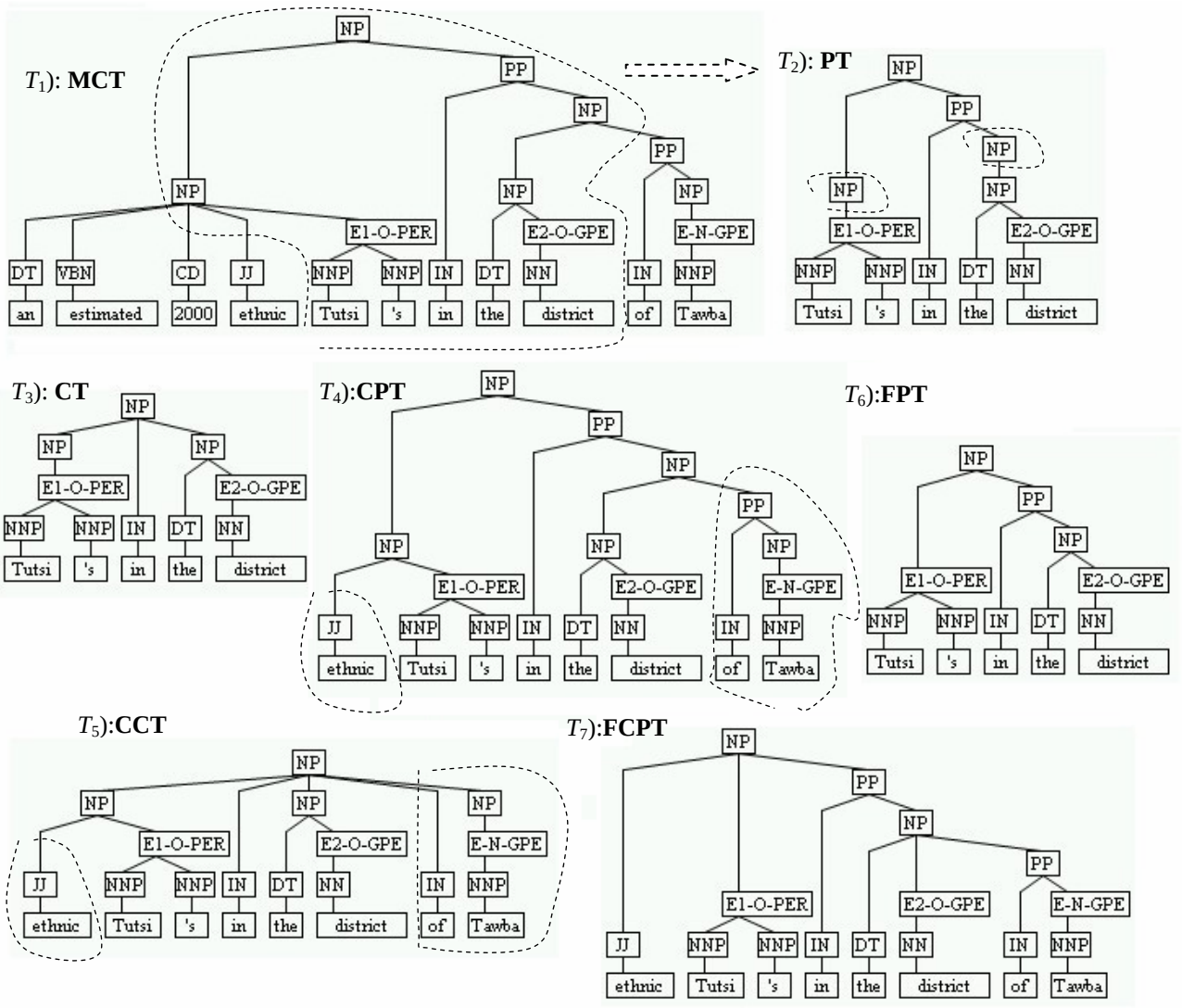


Figure 1. Relation Feature Spaces of the Example Sentence "…… *to stop the merger of an estimated 2000 ethnic* ***Tutsi's*** *in the* ***district*** *of Tawba.*", where the phrase type "E1-O-PER" denotes that the current phrase is the 1st entity, its entity type is "PERSON" and its mention level is "NOMIAL", and likewise for the other two phrase types "E2-O-GPE" and "E-N-GPE".

for relation extraction. T_4 is the **CPT**, where the two structures circled by dashed lines are the so-called context structures. T_5 is the **CCT**, where the additional context structures are also circled by dashed lines. We want to study if the limited context information in the **CPT** and the **CCT** can help boost performance. Moreover, we illustrate the other two flattened trees in T_6 and T_7. The two circled nodes in T_2 are removed in the flattened trees. We want to study if the eliminated small structures are noisy features for relation extraction.

3.2 The Convolution Tree Kernel

Given the relation instances defined in the previous section, we use the same convolution tree kernel as the parse tree kernel (Collins and Duffy, 2001) and the semantic kernel (Moschitti, 2004). Generally, we can represent a parse tree T by a vector of integer counts of each sub-tree type (regardless of its ancestors):

$\phi(T) = (\#\ of\ sub\text{-}trees\ of\ type\ 1, \ldots, \#\ of\ sub\text{-}trees\ of\ type\ i, \ldots, \#\ of\ sub\text{-}trees\ of\ type\ n)$

This results in a very high dimensionality since the number of different sub-trees is exponential in its size. Thus it is computational infeasible to directly use the feature vector $\phi(T)$. To solve the compu-

tational issue, we introduce the tree kernel function which is able to calculate the dot product between the above high dimensional vectors efficiently. The kernel function is defined as follows:

$$K(T_1,T_2) = <\phi(T_1),\phi(T_2)> = \sum_i \phi(T_1)[i],\phi(T_2)[i]$$
$$= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \sum_i I_i(n_1) * I_i(n_2)$$

where N_1 and N_2 are the sets of all nodes in trees T_1 and T_2, respectively, and I_i(n) is the indicator function that is 1 iff a sub-tree of type *i* occurs with root at node n and zero otherwise. Collins and Duffy (2002) show that $K(T_1,T_2)$ is an instance of convolution kernels over tree structures, and which can be computed in $O(|N_1| \times |N_2|)$ by the following recursive definitions (Let $\Delta(n_1,n_2) = \sum_i I_i(n_1) * I_i(n_2)$):

(1) if n_1 and n_2 do not have the same syntactic tag or their children are different then $\Delta(n_1,n_2) = 0$;

(2) else if their children are leaves (POS tags), then $\Delta(n_1,n_2) = 1 \times \lambda$;

(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 number of the children 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 tree sizes.

3.3 Comparison with Previous Work

It would be interesting to review the differences between our method and the feature-based methods. The basic difference between them lies in the relation instance representation and the similarity calculation mechanism. A relation instance in our method is represented as a parse tree while it is represented as a vector of features in the feature-based methods. Our method estimates the similarity between two relation instances by only counting the number of sub-structures that are in common while the feature methods calculate the dot-product between the feature vectors directly. The main difference between them is the different feature spaces. By the kernel method, we implicitly represent a parse tree by a vector of integer counts of each sub-structure type. That is to say, we consider the entire sub-structure types and their occurring frequencies. In this way, on the one hand, the parse tree-related features in the *flat feature set*[2] are embedded in the feature space of our method: "*Base Phrase Chunking*" and "*Parse Tree*" features explicitly appear as substructures of a parse tree. A few of entity-related features in the *flat feature set* are also captured by our feature space: "*entity type*" and "*mention level*" explicitly appear as phrase types in a parse tree. On the other hand, the other features in the *flat feature set*, such as "*word features*", "*bigram word features*", "*overlap*" and "*dependency tree*" are not contained in our feature space. From the syntactic viewpoint, the tree representation in our feature space is more robust than "*Parse Tree Path*" feature in the *flat feature set* since the *path* feature is very sensitive to the small changes of parse trees (Moschitti, 2004) and it also does not maintain the hierarchical information of a parse tree. Due to the extensive exploration of syntactic features by kernel, our method is expected to show better performance than the previous feature-based methods.

It is also worth comparing our method with the previous relation kernels. Since our method only counts the occurrence of each sub-tree without considering its ancestors, our method is not limited by the constraints in Culotta and Sorensen (2004) and that in Bunescu and Mooney (2005) as discussed in Section 2. Compared with Zhao and Grishman's kernel, our method directly uses the original representation of a parse tree while they flatten a parse tree into a *link* and a *path*. Given the above improvements, our method is expected to outperform the previous relation kernels.

4 Experiments

The aim of our experiment is to verify the effectiveness of using richer syntactic structures and the convolution tree kernel for relation extraction.

4.1 Experimental Setting

Corpus: we use the official ACE corpus for 2003 evaluation from LDC as our test corpus. The ACE corpus is gathered from various newspaper, newswire and broadcasts. The same as previous work

[2] For the convenience of discussion, without losing generality, we call the features used in Zhou et al. (2005) and Kambhatla (2004) *flat feature set*.

(Zhou et al., 2005), our experiments are carried out on explicit relations due to the poor inter-annotator agreement in annotation of implicit relations and their limited numbers. The training set consists of 674 annotated text documents and 9683 relation instances. The test set consists of 97 documents and 1386 relation instances. The 2003 evaluation defined 5 types of entities: Persons, Organizations, Locations, Facilities and GPE. Each mention of an entity is associated with a mention type: proper name, nominal or pronoun. They further defined **5** major relation types and **24** subtypes: AT (Base-In, Located…), NEAR (Relative-Location), PART (Part-of, Subsidiary …), ROLE (Member, Owner …) and SOCIAL (Associate, Parent…). As previous work, we explicitly model the argument order of the two mentions involved. We thus model relation extraction as a multi-class classification problem with 10 classes on the major types (2 for each relation major type and a "NONE" class for non-relation (except 1 symmetric type)) and 43 classes on the subtypes (2 for each relation subtype and a "NONE" class for non-relation (except 6 symmetric subtypes)). In this paper, we only measure the performance of relation extraction models on "true" mentions with "true" chaining of coreference (i.e. as annotated by LDC annotators).

Classifier: we select SVM as the classifier used in this paper since SVM can naturally work with kernel methods and it also represents the state-of-the-art machine learning algorithm. We adopt the *one vs. others* strategy and select the one with largest margin as the final answer. The training parameters are chosen using cross-validation (C=2.4 (SVM); λ =0.4(tree kernel)). In our implementation, we use the binary SVMLight developed by Joachims (1998) and Tree Kernel Toolkits developed by Moschitti (2004).

Kernel Normalization: since the size of a parse tree is not constant, we normalize $K(T_1, T_2)$ by dividing it by $\sqrt{K(T_1, T_1) \bullet K(T_2, T_2)}$.

Evaluation Method: we parse the sentence using Charniak parser and iterate over all pair of mentions occurring in the same sentence to generate potential instances. We find the negative samples are 10 times more than the positive samples. Thus data imbalance and sparseness are potential problems. Recall (**R**), Precision (**P**) and F-measure (**F**) are adopted as the performance measure.

4.2 Experimental Results

In order to study the impact of the sole syntactic structure information embedded in parse trees on relation extraction, we remove the entity information from parse trees by replacing the entity-related phrase type ("E1-O-PER", etc., in Figure 1) with "NP". Then we carry out a couple of preliminary experiments on the test set using parse trees regardless of entity information.

Feature Spaces	**P**	**R**	**F**
Minimum Complete Tree	77.45	38.39	51.34
Path-enclosed Tree (PT)	**72.77**	**53.80**	**61.87**
Chunking Tree (**CT**)	75.18	44.75	56.11
Context-Sensitive PT(**CPT**)	77.87	42.80	55.23
Context-Sensitive **CT**	78.33	40.84	53.69
Flattened **PT**	76.86	45.69	57.31
Flattened **CPT**	80.60	41.20	54.53

Table 1. Performance of seven relation feature spaces over the 5 ACE major types using parse tree information only

Table 1 reports the performance of our defined seven relation feature spaces over the 5 ACE major types using parse tree information regardless of any entity information. This preliminary experiments show that:

- Overall the tree kernel over different relation feature spaces is effective for relation extraction since we use the parse tree information only. We will report the detailed performance comparison results between our method and previous work later in this section.
- Using the **PT**s achieves the best performance. This means the portion of a parse tree enclosed by the shortest path between entities can model relations better than other sub-trees.
- Using the **MCT**s get the worst performance. This is because the **MCT**s introduce too much left and right context information, which may be noisy features, as shown in Figure 1. It suggests that only allowing complete (not partial) production rules in the **MCT**s does harm performance.
- The performance of using **CT**s drops by 5 in F-measure compared with that of using the **PT**s. This suggests that the middle and high-level structures beyond chunking is also very useful for relation extraction.

- The context-sensitive trees show lower performance than the corresponding original **PT**s and **CT**s. In some cases (e.g. in sentence "the merge of company A and company B…."; "merge" is the context word), the context information is helpful. However the effective scope of context is hard to determine.
- The two flattened trees perform worse than the original trees, but better than the corresponding context-sensitive trees. This suggests that the removed structures by the flattened trees contribute non-trivial performance improvement.

In the above experiments, the path-enclosed tree displays the best performance among the seven feature spaces when using the parse tree structural information only. In the following incremental experiments, we incorporate more features into the path-enclosed parse trees and it shows significant performance improvement.

Path-enclosed Tree (**PT**)	**P**	**R**	**F**
Parse tree structure information only	72.77	53.80	61.87
+Entity information	76.14	62.85	68.86
+Semantic features	76.32	62.99	69.02

Table 2. Performance of Path-enclosed Trees with different setups over the 5 ACE major types

Table 2 reports the performance over the 5 ACE major types using Path-enclosed trees enhanced with more features in nodes. The 1st row is the baseline performance using structural information only. We then integrate entity information, including Entity type and Mention level features, into the corresponding nodes as shown in Figure 1. The 2nd row in Table 2 reports the performance of this setup. Besides the entity information, we further incorporate the semantic features used in Zhou et al. (2005) into the corresponding leaf nodes. The 3rd row in Table 2 reports the performance of this setup. Please note that in the 2nd and 3rd setups, we still use the same tree kernel function with slight modification on the rule (2) in calculating $\Delta(n_1, n_2)$ (see subsection 3.2) to make it consider more features associated with each individual node: $\Delta(n_1, n_2) = \textit{feature weight} \times \lambda$. From Table 2, we can see that the basic feature of entity information is quite useful, which largely boosts performance by 7 in F-measure. The final performance of our tree kernel method for relation extraction is 76.32/62.99/69.02 in precision/recall/F-measure over the 5 ACE major types.

Methods	**P**	**R**	**F**
Ours: convolution kernel over parse trees	76.32 (64.6)	62.99 (50.76)	69.02 (56.83)
Kambhatla (2004): feature-based ME	- (63.5)	- (45.2)	- (52.8)
Zhou et al. (2005): feature-based SVM	77.2 (63.1)	60.7 (49.5)	68.0 (55.5)
Culotta and Sorensen (2004): dependency kernel	67.1 (-)	35.0 (-)	45.8 (-)
Bunescu and Mooney (2005): shortest path dependency kernel	65.5 (-)	43.8 (-)	52.5 (-)

Table 3. Performance comparison, the numbers in parentheses report the performance over the 24 ACE subtypes while the numbers outside parentheses is for the 5 ACE major types

Table 3 compares the performance of different methods on the ACE corpus[3]. It shows that our method achieves the best-reported performance on both the 24 ACE subtypes and the 5 ACE major types. It also shows that our tree kernel method significantly outperform the previous two dependency kernel algorithms by 16 in F-measure on the 5 ACE relation types[4]. This may be due to two reasons: one reason is that the dependency tree lacks the hierarchical syntactic information, and another reason is due to the two constraints of the two dependency kernels as discussed in Section 2 and Subsection 3.3. The performance improvement by our method suggests that the convolution tree kernel can explore the syntactic features (e.g. parse tree structures and entity information) very effectively and the syntactic features are also particu-

[3] Zhao and Grishman (2005) also evaluated their algorithm on the ACE corpus and got good performance. But their experimental data is for 2004 evaluation, which defined 7 entity types with 44 entity subtypes, and 7 relation major types with 27 subtypes, so we are not ready to compare with each other.

[4] Bunescu and Mooney (2005) used the ACE 2002 corpus, including 422 documents, which is known to have many inconsistencies than the 2003 version. Culotta and Sorensen (2004) used an ACE corpus including about 800 documents, and they did not specify the corpus version. Since the testing corpora are in different sizes and versions, strictly speaking, it is not ready to compare these methods exactly and fairly. Thus Table 3 is only for reference purpose. We just hope that we can get a few clues from this table.

larly effective for the task of relation extraction. In addition, we observe from Table 1 that the feature space selection (the effective portion of a parse tree) is also critical to relation extraction.

Error Type	# of error instance
False Negative	414
False Positive	173
Cross Type	97

Table 4. Error Distribution

Finally, Table 4 reports the error distribution in the case of the 3rd experiment in Table 2. It shows that 85.9% (587/684) of the errors result from relation detection and only 14.1% (97/684) of the errors result from relation characterization. This is mainly due to the imbalance of the positive/negative instances and the sparseness of some relation types on the ACE corpus.

5 Conclusion and Future Work

In this paper, we explore the syntactic features using convolution tree kernels for relation extraction. We conclude that: 1) the relations between entities can be well represented by parse trees with carefully calibrating effective portions of parse trees; 2) the syntactic features embedded in a parse tree are particularly effective for relation extraction; 3) the convolution tree kernel can effectively capture the syntactic features for relation extraction.

The most immediate extension of our work is to improve the accuracy of relation detection. We may adopt a two-step method (Culotta and Sorensen, 2004) to separately model the relation detection and characterization issues. We may integrate more features (such as head words or WordNet semantics) into nodes of parse trees. We can also benefit from the learning algorithm to study how to solve the data imbalance and sparseness issues from the learning algorithm viewpoint. In the future, we would like to test our algorithm on the other version of the ACE corpus and to develop fast algorithm (Vishwanathan and Smola, 2002) to speed up the training and testing process of convolution kernels.

Acknowledgements: We would like to thank Dr. Alessandro Moschitti for his great help in using his Tree Kernel Toolkits and fine-tuning the system. We also would like to thank the three anonymous reviewers for their invaluable suggestions.

References

ACE. 2004. *The Automatic Content Extraction (ACE) Projects*. http://www.ldc.upenn.edu/Projects/ACE/

Bunescu R. C. and Mooney R. J. 2005. *A Shortest Path Dependency Kernel for Relation Extraction*. EMNLP-2005

Charniak E. 2001. Immediate-head Parsing for Language Models. ACL-2001

Collins M. and Duffy N. 2001. *Convolution Kernels for Natural Language*. NIPS-2001

Culotta A. and Sorensen J. 2004. *Dependency Tree Kernel for Relation Extraction*. ACL-2004

Haussler D. 1999. *Convolution Kernels on Discrete Structures.* Technical Report UCS-CRL-99-10, University of California, Santa Cruz.

Joachims T. 1998. *Text Categorization with Support Vecor Machine: learning with many relevant features*. ECML-1998

Kambhatla Nanda. 2004. *Combining lexical, syntactic and semantic features with Maximum Entropy models for extracting relations*. ACL-2004 (poster)

Lodhi H., Saunders C., Shawe-Taylor J., Cristianini N. and Watkins C. 2002. *Text classification using string kernel*. Journal of Machine Learning Research, 2002(2):419-444

Miller S., Fox H., Ramshaw L. and Weischedel R. 2000. *A novel use of statistical parsing to extract information from text*. NAACL-2000

Moschitti Alessandro. 2004. *A Study on Convolution Kernels for Shallow Semantic Parsing*. ACL-2004

MUC. 1987-1998. The nist MUC website: http://www.itl.nist.gov/iaui/894.02/related_projects/muc/

Suzuki J., Hirao T., Sasaki Y. and Maeda E. 2003. *Hierarchical Directed Acyclic Graph Kernel: Methods for Structured Natural Language Data*. ACL-2003

Vishwanathan S.V.N. and Smola A.J. 2002. *Fast kernels for String and Tree Matching*. NIPS-2002

Zelenko D., Aone C. and Richardella A. 2003. *Kernel Methods for Relation Extraction*. Journal of Machine Learning Research. 2003(2):1083-1106

Zhao Shubin and Grishman Ralph. 2005. *Extracting Relations with Integrated Information Using Kernel Methods*. ACL-2005

Zhou Guodong, Su Jian, Zhang Jie and Zhang Min. 2005. *Exploring Various Knowledge in Relation Extraction*. ACL-2005

Integrating Probabilistic Extraction Models and Data Mining to Discover Relations and Patterns in Text

Aron Culotta
University of Massachusetts
Amherst, MA 01003
culotta@cs.umass.edu

Andrew McCallum
University of Massachusetts
Amherst, MA 01003
mccallum@cs.umass.edu

Jonathan Betz
Google, Inc.
New York, NY 10018
jtb@google.com

Abstract

In order for relation extraction systems to obtain human-level performance, they must be able to incorporate relational patterns inherent in the data (for example, that one's sister is likely one's mother's daughter, or that children are likely to attend the same college as their parents). Hand-coding such knowledge can be time-consuming and inadequate. Additionally, there may exist many interesting, unknown relational patterns that both improve extraction performance and provide insight into text. We describe a probabilistic extraction model that provides mutual benefits to both "top-down" relational pattern discovery and "bottom-up" relation extraction.

1 Introduction

Consider these four sentences:

1. George W. Bush's father is George H. W. Bush.
2. George H. W. Bush's sister is Nancy Bush Ellis.
3. Nancy Bush Ellis's son is John Prescott Ellis.
4. John Prescott Ellis analyzed George W. Bush's campaign.

We would like to build an automated system to extract the set of relations shown in Figure 1.

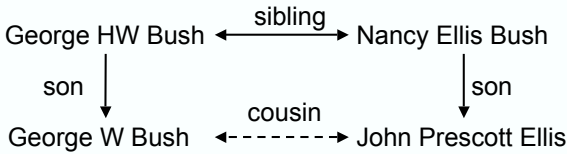


Figure 1: Bush family tree

State of the art extraction algorithms may be able to detect the *son* and *sibling* relations from local language clues. However, the *cousin* relation is only *implied* by the text and requires additional knowledge to be extracted. Specifically, the system requires knowledge of familial relation patterns.

One could imagine a system that accepts such rules as input (e.g. *cousin = father's sister's son*) and applies them to extract implicit relations. However, exhaustively enumerating all possible rules can be tedious and incomplete. More importantly, many relational patterns unknown *a priori* may both improve extraction accuracy and uncover informative trends in the data (e.g. that children often adopt the religion of their parents). Indeed, the goal of data mining is to learn such patterns from database regularities. Since these patterns will not always hold, we would like to handle them probabilistically.

We propose an integrated supervised machine learning method that learns both contextual and relational patterns to extract relations. In particular, we construct a linear-chain conditional random field (Lafferty et al., 2001; Sutton and McCallum, 2006) to extract relations from biographical texts while simultaneously discovering interesting relational patterns that improve extraction performance.

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 296–303,
New York, June 2006.

2 Related Work

This work can be viewed as a step toward the integration of information extraction and data mining technology, a direction of growing interest. Nahm and Mooney (2000) present a system that mines association rules from a database constructed from automatically extracted data, then applies these learned rules to improve data field recall without revisiting the text. Our work attempts to more tightly integrate the extraction and mining tasks by learning relational patterns that can be included probabilistically into extraction to improve its accuracy; also, our work focuses on mining from relational graphs, rather than single-table databases.

McCallum and Jensen (2003) argue the theoretical benefits of an integrated probabilistic model for extraction and mining, but do not construct such a system. Our work is a step in the direction of their proposal, using an inference procedure based on a closed-loop iteration between extraction and relational pattern discovery.

Most other work in this area mines raw text, rather than a database automatically populated via extraction (Hearst, 1999; Craven et al., 1998).

This work can also be viewed as part of a trend to perform joint inference across multiple language processing tasks (Miller et al., 2000; Roth and tau Yih, 2002; Sutton and McCallum, 2004).

Finally, using relational paths between entities is also examined in (Richards and Mooney, 1992) to escape local maxima in a first-order learning system.

3 Relation Extraction as Sequence Labeling

Relation extraction is the task of discovering semantic connections between entities. In text, this usually amounts to examining pairs of entities in a document and determining (from local language cues) whether a relation exists between them. Common approaches to this problem include pattern matching (Brin, 1998; Agichtein and Gravano, 2000), kernel methods (Zelenko et al., 2003; Culotta and Sorensen, 2004; Bunescu and Mooney, 2006), logistic regression (Kambhatla, 2004), and augmented parsing (Miller et al., 2000).

The pairwise classification approach of kernel methods and logistic regression is commonly a two-phase method: first the entities in a document are identified, then a relation type is predicted for each pair of entities. This approach presents at least two difficulties: (1) enumerating all pairs of entities, even when restricted to pairs within a sentence, results in a low density of positive relation examples; and (2) errors in the entity recognition phase can propagate to errors in the relation classification stage. As an example of the latter difficulty, if a person is mislabeled as a company, then the relation classifier will be unsuccessful in finding a *brother* relation, despite local evidence.

We avoid these difficulties by restricting our investigation to *biographical texts*, e.g. encyclopedia articles. A biographical text mostly discusses one entity, which we refer to as the *principal entity*. We refer to other mentioned entities as *secondary entities*. For each secondary entity, our goal is to predict what relation, if any, it has to the principal entity.

This formulation allows us to treat relation extraction as a *sequence labeling* task such as *named-entity recognition* or *part-of-speech tagging*, and we can now apply models that have been successful on those tasks. By anchoring one argument of relations to be the principal entity, we alleviate the difficulty of enumerating all pairs of entities in a document. By converting to a sequence labeling task, we fold the entity recognition step into the relation extraction task. There is no initial pass to label each entity as a person or company. Instead, an entity's label is its relation to the principal entity. Below is an example of a labeled article:

George W. Bush

George is the son of $\underbrace{\text{George H. W. Bush}}_{\textbf{father}}$ and $\underbrace{\text{Barbara Bush}}_{\textbf{mother}}$.

Additionally, by using a sequence model we can capture the dependence between adjacent labels. For example, in our data it is common to see phrases such as "son of the Republican president George H. W. Bush" for which the labels *politicalParty*, *jobTitle*, and *father* occur consecutively. Sequence models are specifically designed to handle these kinds of dependencies. We now discuss the details of our extraction model.

3.1 Conditional Random Fields

We build a model to extract relations using linear-chain conditional random fields (CRFs) (Lafferty et al., 2001; Sutton and McCallum, 2006). CRFs are undirected graphical models (i.e. Markov networks) that are discriminatively-trained to maximize the conditional probability of a set of output variables $\mathbf{y}$ given a set of input variables $\mathbf{x}$. This conditional distribution has the form

$$p_\Lambda(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_\mathbf{x}} \prod_{c \in C} \phi_c(\mathbf{y}_c, \mathbf{x}_c; \Lambda) \qquad (1)$$

where ϕ are potential functions parameterized by Λ and $Z_x = \sum_\mathbf{y} \prod_{c \in C} \phi(\mathbf{y}_c, \mathbf{x}_c)$ is a normalization factor. Assuming ϕ_c factorizes as a log-linear combination of arbitrary features computed over clique c, then $\phi_c(\mathbf{y}_c, \mathbf{x}_c; \Lambda) = \exp\left(\sum_k \lambda_k f_k(\mathbf{y}_c, \mathbf{x}_c)\right)$, where f is a set of arbitrary *feature functions* over the input, each of which has an associate model parameter λ_k. Parameters $\Lambda = \{\lambda_k\}$ are a set of real-valued weights typically estimated from labeled training data by maximizing the data likelihood function using gradient ascent.

In these experiments, we make a first-order Markov assumption on the dependencies among $\mathbf{y}$, resulting in a linear-chain CRF.

4 Relational Patterns

The modeling flexibility of CRFs permits the feature functions to be complex, overlapping features of the input without requiring additional assumptions on their inter-dependencies. In addition to common language features (e.g. neighboring words and syntactic information), in this work we explore features that cull relational patterns from a database of entities.

As described in the introductory example (Figure 1), context alone is often insufficient to extract relations. Even in simpler examples, it may be the case that modeling relational patterns can improve extraction accuracy.

To capture this evidence, we compute features from a database to indicate relational connections between entities, similar to the *relational path-finding* performed in Richards and Mooney (1992).

Imagine that the four sentence example about the Bush family is included in a training set, and the entities are labeled with their correct relations. In this case, the *cousin* relation in sentence 4 would also be labeled. From this data, we can create a relational database that contains the relations in Figure 1.

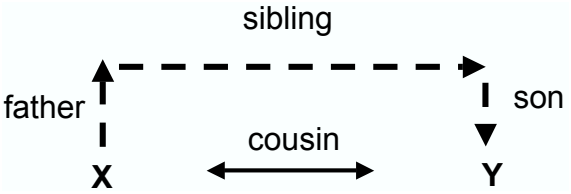


Figure 2: A feature template for the *cousin* relation.

Assume sentence 4 comes from a biography about John Ellis. We calculate a feature for the entity George W. Bush that indicates the path from John Ellis to George W. Bush in the database, annotating each edge in the path with its relation label; i.e. *father-sibling-son*. By abstracting away the actual entity names, we have created a *cousin* template feature, as shown in Figure 2.

By adding these relational paths as features to the model, we can learn interesting relational patterns that may have low precision (e.g. "people are likely to be friends with their classmates") without hampering extraction performance. This is in contrast to the system described in Nahm and Mooney (2000), in which patterns are induced from a noisy database and then applied directly to extraction. In our system, since each learned path has an associated weight, it is simply another piece of evidence to help the extractor. Low precision patterns may have lower weights than high precision patterns, but they will still influence the extractor.

A nice property of this approach is that examining highly weighted patterns can provide insight into regularities of the data.

4.1 Feature Induction

During CRF training, weights are learned for each relational pattern. Patterns that increase extraction performance will receive higher weights, while patterns that have little effect on performance will receive low weights.

We can explore the space of possible conjunctions of these patterns using feature induction for CRFs, as described in McCallum (2003). Search through the large space of possible conjunctions is guided

by adding features that are estimated to increase the likelihood function most.

When feature induction is used with relational patterns, we can view this as a type of data mining, in which patterns are created based on their influence on an extraction model. This is similar to work by Dehaspe (1997), where *inductive logic programming* is embedded as a feature induction technique for a maximum entropy classifier. Our work restricts induced features to conjunctions of base features, rather than using first-order clauses. However, the patterns we learn are based on information extracted from natural language.

4.2 Iterative Database Construction

The top-down knowledge provided by data mining algorithms has the potential to improve the performance of information extraction systems. Conversely, bottom-up knowledge generated by extraction systems can be used to populate a large database, from which more top-down knowledge can be discovered. By carefully communicating the uncertainty between these systems, we hope to iteratively expand a knowledge base, while minimizing fallacious inferences.

In this work, the top-down knowledge consists of relational patterns describing the database path between entities in text. The uncertainty of this knowledge is handled by associating a real-valued CRF weight with each pattern, which increases when the pattern is predictive of other relations. Thus, the extraction model can adapt to noise in these patterns.

Since we also desire to extract relations between entities that appear in text but not in the database, we first populate the database with relations extracted by a CRF that does not use relational patterns. We then do further extraction with a CRF that incorporates the relational patterns found in this automatically generated database. In this manner, we create a closed-loop system that alternates between bottom-up extraction and top-down pattern discovery. This approach can be viewed as a type of alternating optimization, with analogies to formal methods such as expectation-maximization.

The uncertainty in the bottom-up extraction step is handled by estimating the confidence of each extraction and pruning the database to remove entries with low confidence. One of the benefits of a probabilistic extraction model is that confidence estimates can be straight-forwardly obtained. Culotta and McCallum (2004) describe the *constrained forward-backward* algorithm to efficiently estimate the conditional probability that a segment of text is correctly extracted by a CRF.

Using this algorithm, we associate a confidence value with each relation extracted by the CRF. This confidence value is then used to limit the noise introduced by incorrect extractions. This differs from Nahm and Mooney (2000) and Mooney and Bunescu (2005), in which standard decision tree rule learners are applied to the unfiltered output of extraction.

4.3 Extracting Implicit Relations

An *implicit relation* is one that does not have direct contextual evidence, for example the *cousin* relation in our initial example. Implicit relations generally require some background knowledge to be detected, such as relational patterns (e.g. rules about familial relations). These are the sorts of relations on which current extraction models perform most poorly.

Notably, these are exactly the sorts of relations that are likely to have the biggest impact on information access. A system that can accurately discover knowledge that is only implied by the text will dramatically increase the amount of information a user can uncover, effectively providing access to the *implications* of a corpus.

We argue that integrating top-down and bottom-up knowledge discovery algorithms discussed in Section 4.2 can enable this technology. By performing pattern discovery in conjunction with information extraction, we can collate facts from multiple sources to infer new relations. This is an example of *cross-document fusion* or *cross-document information extraction*, a growing area of research transforming raw extractions into usable knowledge bases (Mann and Yarowsky, 2005; Masterson and Kushmerik, 2003).

5 Experiments

5.1 Data

We sampled 1127 paragraphs from 271 articles from the online encyclopedia Wikipedia[1] and labeled a to-

[1] http://www.wikipedia.org

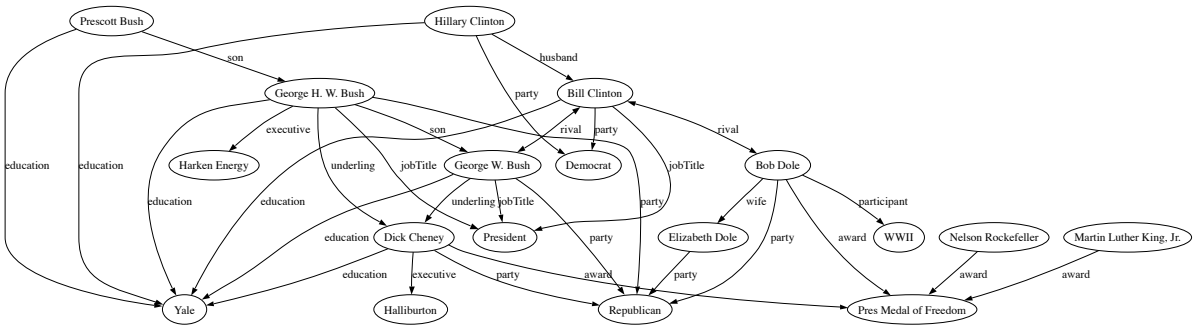


Figure 3: An example of the connectivity of the entities in the data.

birthday	birth year	death day
death year	nationality	visited
birth place	death place	religion
job title	member of	cousin
friend	discovered	education
employer	associate	opus
participant	influence	award
brother	wife	supported idea
executive of	political party	supported person
founder	son	father
rival	underling	superior
role	inventor	husband
grandfather	sister	brother-in-law
nephew	mother	daughter
granddaughter	grandson	great-grandson
grandmother	rival organization	owner of
uncle	descendant	ancestor
great-grandfather	aunt	

Table 1: The set of labeled relations.

tal of 4701 relation instances. In addition to a large set of person-to-person relations, we also included links between people and organizations, as well as biographical facts such as *birthday* and *jobTitle*. In all, there are 53 labels in the training data (Table 1).

We sample articles that result in a high density of interesting relations by choosing, for example, a collection of related family members and associates. Figure 3 shows a small example of the type of connections in the data. We then split the data into training and testing sets (70-30 split), attempting to separate the entities into connected components. For example, all Bush family members were placed in the training set, while all Kennedy family members were placed in the testing set. While there are still occasional paths connecting entities in the training set to those in the test set, we believe this methodology reflects a typical real-world scenario in which we would like to extend an existing database to a different, but slightly related, domain.

The structure of the Wikipedia articles somewhat simplifies the extraction task, since important entities are hyper-linked within the text. This provides an automated way to detect entities in the text, although these entities are not classified by type. This also allows us to easily construct database queries, since we can reason at the entity level, rather than the token level. (Although, see Sarawagi and Cohen (2004) for extensions of CRFs that model the entity length distribution.) The results we report here are constrained to predict relations only for hyper-linked entities. Note that despite this property, we still desire to use a sequence model to capture the dependencies between adjacent labels.

We use the MALLET CRF implementation (McCallum, 2002) with the default regularization parameters.

Based on initial experiments, we restrict relational path features to length two or three. Paths of length one will learn trivial paths and can lead to overfitting. Paths longer than three can increase computational costs without adding much new information.

In addition to the relational pattern features described in Section 4, the list of local features includes **context words** (such as the token identity within a 6 word window of the target token), **lexicons** (such as whether a token appears in a list of cities, people, or companies), **regular expressions** (such as whether the token is capitalized or contains digits or punctuation), **part-of-speech** (predicted by a CRF that was trained separately for part of speech tagging), **prefix/suffix** (such as whether a word ends in *-ed* or begins with *ch-*), and **offset conjunctions** (combinations of adjacent features within a window of size six).

	ME	CRF_0	CRF_r	$CRF_r 0.9$	$CRF_r 0.5$	CRF_t	$CRF_t 0.5$
F1	.5489	.5995	.6100	.6008	**.6136**	.6791	.6363
P	.6475	.7019	.6799	**.7177**	.7095	.7553	.7343
R	.4763	.5232	**.5531**	.5166	.5406	.6169	.5614

Table 2: Results comparing the relative benefits of using relational patterns in extraction.

5.2 Extraction Results

We evaluate performance by calculating the precision (**P**) and recall (**R**) of extracted relations, as well as the **F1** measure, which is the harmonic mean of precision and recall.

CRF_0 is the conditional random field constructed without relational features. Results for CRF_0 are displayed in the second column of Table 2. *ME* is a maximum entropy classifier trained on the same feature set as CRF_0. The difference between these two models is that CRF_0 models the dependence of relations that appear consecutively in the text. The superior performance of CRF_0 suggests that this dependence is important to capture.

The remaining models incorporate the relational patterns described in Section 4. We compare three different confidence thresholds for the construction of the initial testing database, as described in Section 4.2. CRF_r uses no threshold, while $CRF_r 0.9$ and $CRF_r 0.5$ restrict the database to extractions with confidence greater than 0.9 and 0.5, respectively.

As shown by comparing CRF_0 and CRF_r in Table 2, the relational features constructed from the database with no confidence threshold provides a considerable boost in recall (reducing error by 7%), at the cost of a decrease in precision. Here we see the effect of making fallacious inferences on a noisy database.

In column four, we see the opposite effect for the overly conservative threshold of $CRF_r 0.9$. Here, precision improves slightly over CRF_0, and considerably over CRF_r (12% error reduction), but this is accompanied by a drop in recall (8% reduction).

Finally, in column five, a confidence of 0.5 results in the best F1 measure (a 3.5% error reduction over CRF_0). $CRF_r 0.5$ also obtains better recall and precision than CRF_0, reducing recall error by 3.6%, precision error by 2.5%.

Comparing the performance on different relation types, we find that the biggest increase from CRF_0 to $CRF_r 0.5$ is on the *memberOf* relation, for which the F1 score improves from 0.4211 to 0.6093. We conjecture that the reason for this is that the patterns most useful for the *memberOf* label contain relations that are well-detected by the first-pass CRF. Also, the local language context seems inadequate to properly extract this relation, given the low performance of CRF_0.

To better gauge how much relational pattern features are affected by errors in the database, we run two additional experiments for which the relational features are fixed to be correct. That is, imagine that we construct a database from the true labeling of the testing data, and create the relational pattern features from this database. Note that this does not trivialize the problem, since there are no relational path features of length one (e.g., if X is the wife of Y, there will be no feature indicating this).

We construct two experiments under this scheme, one where the entire test database is used (CRF_t), and another where only half the relations are included in the test database, selected uniformly at random ($CRF_t 0.5$).

Column six shows the improvements enabled by using the complete testing database. More interestingly, column seven shows that even with only half the database accurately known, performance improves considerably over both *CRF* and $CRF_r 0.5$. A realistic scenario for $CRF_t 0.5$ is a semi-automated system, in which a partially-filled database is used to bootstrap extraction.

5.3 Mining Results

Comparing the impact of discovered patterns on extraction is a way to objectively measure mining performance. We now give a brief subjective evaluation of the learned patterns. By examining relational patterns with high weights for a particular label, we can glean some regularities from our dataset. Examples of such patterns are in Table 3.

Relation	Relational Path Feature
mother	father → wife
cousin	mother → husband → nephew
friend	education → student
education	father → education
boss	boss → son
memberOf	grandfather → memberOf
rival	politicalParty → member → rival

Table 3: Examples of highly weighted relational patterns.

From the familial relations in our training data, we are able to discover many equivalences for mothers, cousins, grandfathers, and husbands. In addition to these high precision patterns, the system also generates interesting, low precision patterns. Row 3-7 of Table 3 can be summarized by the following generalizations: friends tend to be classmates; children of alumni often attend the same school as their parents; a boss' child often becomes the boss; grandchildren are often members of the same organizations as their grandparents; and rivals of a person from one political party are often rivals of other members of the same political party. While many of these patterns reflect the high concentration of political entities and familial relations in our training database, many will have applicability across domains.

5.4 Implicit Relations

It is difficult to measure system performance on implicit relations, since our labeled data does not distinguish between explicit and implicit relations. Additionally, accurately labeling all implicit relations is challenging even for a human annotator.

We perform a simple exploratory analysis to determine how relational patterns can help discover implicit relations. We construct a small set of synthetic sentences for which CRF_0 successfully extracts relations using contextual features. We then add sentences with slightly more ambiguous language and measure whether CRF_r can overcome this ambiguity using relational pattern features.

For example, we create an article about an entity named "Bob Smith" that includes the sentences "His brother, Bill Smith, was a biologist" and "His *companion*, Bill Smith, was a biologist." CRF_0 successfully returns the brother relation in the first sentence, but not the second. After a fact is added to the database that says Bob and Bill have a brother in common named John, CRF_r is able to correctly label the second sentence in spite of the ambiguous word "companion," because CRF_0 has a highly-weighted relational pattern feature for brother.

Similar behavior is observed for low precision patterns like "associates tend to win the same awards." A synthetic article for the entity "Tom Jones" contains the sentences "He was awarded the Pulitzer Prize in 1998" and "Tom got the Pulitzer Prize in 1998." Because CRF_0 is highly-reliant on the presence of the verb "awarded" or "won" to indicate a prize fact, it fails to label the second sentence correctly. After the database is augmented to include the fact that Tom's associate Jill received the Pulitzer Prize, CRF_r labels the second sentence correctly.

However, we also observed that CRF_r still requires some contextual clues to extract implicit relations. For example, if the Tom Jones article instead contains the sentence "The Pulitzer Prize was awarded to him in 1998," neither CRF labels the prize fact correctly, since this passive construction is rarely seen in the training data.

We conclude from this brief analysis that relational patterns used by CRF_r can help extract implicit relations when (1) the database contains accurate relational information, and (2) the sentence contains limited contextual clues. Since relational patterns are treated only as additional features by CRF_r, they are generally not powerful enough to overcome a complete absence of contextual clues. From this perspective, relational patterns can be seen as enhancing the signal from contextual clues. This differs from deterministically applying learned rules independent of context, which may boost recall at the cost of precision.

6 Conclusions and Future Work

We have shown that integrating pattern discovery with relation extraction can lead to improved performance on each task.

In the future, we wish to explore extending this methods to larger datasets, where we expect relational patterns to be even more interesting. Also, we plan to improve upon iterative database construction by performing joint inference among distant

relations in an article. Inference in these highly-connected models will likely require approximate methods. Additionally, we wish to focus on extracting implicit relations, dealing more formally with the precision-recall trade-off inherent in applying noisy rules to improve extraction.

7 Acknowledgments

Thanks to the Google internship program, and to Charles Sutton for providing the CRF POS tagger. This work was supported in part by the Center for Intelligent Information Retrieval, in part by U.S. Government contract #NBCH040171 through a subcontract with BBNT Solutions LLC, in part by The Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grant #IIS-0326249, and in part by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under contract number NBCHD030010. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM International Conference on Digital Libraries*.

Sergey Brin. 1998. Extracting patterns and relations from the world wide web. In *WebDB Workshop at 6th International Conference on Extending Database Technology*.

Razvan Bunescu and Raymond Mooney. 2006. Subsequence kernels for relation extraction. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*. MIT Press, Cambridge, MA.

Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew K. McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. 1998. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence*, pages 509–516, Madison, US. AAAI Press, Menlo Park, US.

Aron Culotta and Andrew McCallum. 2004. Confidence estimation for information extraction. In *Human Langauge Technology Conference (HLT 2004)*, Boston, MA.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *ACL*.

L. Dehaspe. 1997. Maximum entropy modeling with clausal constraints. In *Proceedings of the Seventh International Workshop on Inductive Logic Programming*, pages 109–125, Prague, Czech Republic.

M. Hearst. 1999. Untangling text data mining. In *37th Annual Meeting of the Association for Computational Linguistics*.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *ACL*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA.

Gideon Mann and David Yarowsky. 2005. Multi-field information extraction and cross-document fusion. In *ACL*.

D. Masterson and N. Kushmerik. 2003. Information extraction from multi-document threads. In *ECML-2003: Workshop on Adaptive Text Extraction and Mining*, pages 34–41.

Andrew McCallum and David Jensen. 2003. A note on the unification of information extraction and data mining using conditional-probability, relational models. In *IJCAI03 Workshop on Learning Statistical Models from Relational Data*.

Andrew McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Andrew McCallum. 2003. Efficiently inducing features of conditional random fields. In *Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI03)*.

Scott Miller, Heidi Fox, Lance A. Ramshaw, and Ralph Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *ANLP*.

Raymond J. Mooney and Razvan Bunescu. 2005. Mining knowledge from text using information extraction. *SigKDD Explorations on Text Mining and Natural Language Processing*.

Un Yong Nahm and Raymond J. Mooney. 2000. A mutually beneficial integration of data mining and information extraction. In *AAAI/IAAI*.

Bradley L. Richards and Raymond J. Mooney. 1992. Learning relations by pathfinding. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 50–55, San Jose, CA.

Dan Roth and Wen tau Yih. 2002. Probabilistic reasoning for entity and relation recognition. In *COLING*.

Sunita Sarawagi and William W. Cohen. 2004. Semi-markov conditional random fields for information extraction. In *NIPS 04*.

Charles Sutton and Andrew McCallum. 2004. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML)*.

Charles Sutton and Andrew McCallum. 2006. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press. To appear.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106.

Preemptive Information Extraction using Unrestricted Relation Discovery

Yusuke Shinyama **Satoshi Sekine**

New York University
715, Broadway, 7th Floor
New York, NY, 10003
{yusuke,sekine}@cs.nyu.edu

Abstract

We are trying to extend the boundary of Information Extraction (IE) systems. Existing IE systems require a lot of time and human effort to tune for a new scenario. Preemptive Information Extraction is an attempt to automatically create all feasible IE systems in advance without human intervention. We propose a technique called Unrestricted Relation Discovery that discovers all possible relations from texts and presents them as tables. We present a preliminary system that obtains reasonably good results.

1 Background

Every day, a large number of news articles are created and reported, many of which are unique. But certain types of events, such as hurricanes or murders, are reported again and again throughout a year. The goal of Information Extraction, or IE, is to retrieve a certain type of news event from past articles and present the events as a table whose columns are filled with a name of a person or company, according to its role in the event. However, existing IE techniques require a lot of human labor. First, you have to specify the type of information you want and collect articles that include this information. Then, you have to analyze the articles and manually craft a set of patterns to capture these events. Most existing IE research focuses on reducing this burden by helping people create such patterns. But each time you want to extract a different kind of information, you need to repeat the whole process: specify articles and adjust its patterns, either manually or semi-automatically. There is a bit of a dangerous pitfall here. First, it is hard to estimate how good the system can be after months of work. Furthermore, you might not know if the task is even doable in the first place. Knowing what kind of information is easily obtained in advance would help reduce this risk.

An IE task can be defined as finding a relation among several entities involved in a certain type of event. For example, in the MUC-6 management succession scenario, one seeks a relation between COMPANY, PERSON and POST involved with hiring/firing events. For each row of an extracted table, you can always read it as "COMPANY hired (or fired) PERSON for POST." The relation between these entities is retained throughout the table. There are many existing works on obtaining extraction patterns for pre-defined relations (Riloff, 1996; Yangarber et al., 2000; Agichtein and Gravano, 2000; Sudo et al., 2003).

Unrestricted Relation Discovery is a technique to automatically discover such relations that repeatedly appear in a corpus and present them as a table, with absolutely no human intervention. Unlike most existing IE research, a user does not specify the type of articles or information wanted. Instead, a system tries to find all the kinds of relations that are reported multiple times and can be reported in tabular form. This technique will open up the possibility of trying new IE scenarios. Furthermore, the system itself can be used as an IE system, since an obtained relation is already presented as a table. If this system works to a certain extent, tuning an IE system becomes a search problem: all the tables are already built "preemptively." A user only needs to search for a relevant table.

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 304–311,
New York, June 2006.

Article	dump	be-hit
2005-09-23	Katrina	New Orleans
2005-10-02	Longwang	Taiwan
2005-11-20	Gamma	Florida

Keywords: storm, evacuate, coast, rain, hurricane

Table 1: Sample discovered relation.

We implemented a preliminary system for this technique and obtained reasonably good performance. Table 1 is a sample relation that was extracted as a table by our system. The columns of the table show article dates, names of hurricanes and the places they affected respectively. The headers of the table and its keywords were also extracted automatically.

2 Basic Idea

In Unrestricted Relation Discovery, the discovery process (i.e. creating new tables) can be formulated as a clustering task. The key idea is to cluster a set of articles that contain entities bearing a similar relation to each other in such a way that we can construct a table where the entities that play the same role are placed in the same column.

Suppose that there are two articles A and B, and both report hurricane-related news. Article A contains two entities "Katrina" and "New Orleans", and article B contains "Longwang" and "Taiwan". These entities are recognized by a Named Entity (NE) tagger. We want to discover a relation among them. First, we introduce a notion called "basic pattern" to form a relation. A basic pattern is a part of the text that is syntactically connected to an entity. Some examples are "`X is hit`" or "`Y's residents`". Figure 1 shows several basic patterns connected to the entities "Katrina" and "New Orleans" in article A. Similarly, we obtain the basic patterns for article B. Now, in Figure 2, both entities "Katrina" and "Longwang" have the basic pattern "`headed`" in common. In this case, we connect these two entities to each other. Furthermore, there is also a common basic pattern "`was-hit`" shared by "New Orleans" and "Taiwan". Now, we found two sets of entities that can be placed in correspondence at the same time. What does this mean? We can infer that both entity sets ("Katrina"-"New Orleans" and "Longwang"-"Taiwan") represent a certain relation that has something in common: *a hurricane name and the place it affected*. By finding multiple parallel correspondences between two articles, we can estimate the similarity of their relations.

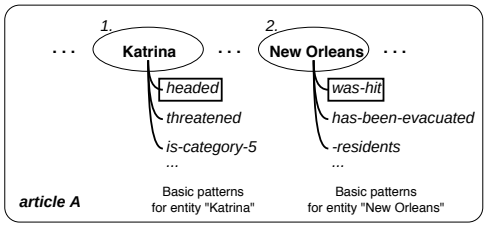
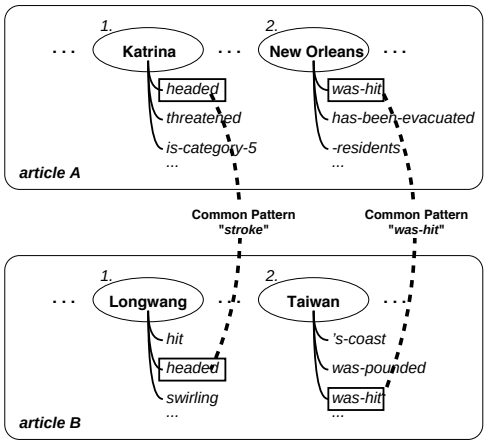


Figure 1: Obtaining basic patterns.



Obtained Table

article A	Katrina	New Orleans
article B	Longwang	Taiwan

Figure 2: Finding a similar relation from two articles.

Generally, in a clustering task, one groups items by finding similar pairs. After finding a pair of articles that have a similar relation, we can bring them into the same cluster. In this case, we cluster articles by using their basic patterns as features. However, each basic pattern is still connected to its entity so that we can extract the name from it. We can consider a basic pattern to represent something like the "role" of its entity. In this example, the entities that had "`headed`" as a basic pattern are *hurricanes*, and the entities that had "`was-hit`" as a basic pattern are *the places it affected*. By using basic patterns, we can align the entities into the corresponding column that represents a certain role in the relation. From this example, we create a two-by-two table, where each column represents the roles of the entities, and each row represents a different article, as shown in the bottom of Figure 2.

We can extend this table by finding another article

in the same manner. In this way, we gradually extend a table while retaining a relation among its columns. In this example, the obtained table is just what an IE system (whose task is to find a hurricane name and the affected place) would create.

However, these articles might also include other things, which could represent different relations. For example, the governments might call for help or some casualties might have been reported. To obtain such relations, we need to choose different entities from the articles. Several existing works have tried to extract a certain type of relation by manually choosing different pairs of entities (Brin, 1998; Ravichandran and Hovy, 2002). Hasegawa et al. (2004) tried to extract multiple relations by choosing entity types. We assume that we can find such relations by trying all possible combinations from a set of entities we have chosen in advance; some combinations might represent a hurricane and government relation, and others might represent a place and its casualties. To ensure that an article can have several different relations, we let each article belong to several different clusters.

In a real-world situation, only using basic patterns sometimes gives undesired results. For example, "*(President) Bush flew to Texas*" and "*(Hurricane) Katrina flew to New Orleans*" both have a basic pattern "`flew to`" in common, so "Bush" and "Katrina" would be put into the same column. But we want to separate them in different tables. To alleviate this problem, we put an additional restriction on clustering. We use a bag-of-words approach to discriminate two articles: if the word-based similarity between two articles is too small, we do not bring them together into the same cluster (i.e. table). We exclude names from the similarity calculation at this stage because we want to link articles about the same type of event, not the same instance. In addition, we use the frequency of each basic pattern to compute the similarity of relations, since basic patterns like "say" or "have" appear in almost every article and it is dangerous to rely on such expressions.

Increasing Basic Patterns

In the above explanation, we have assumed that we can obtain enough basic patterns from an article. However, the actual number of basic patterns that one can find from a single article is usually not enough, because the number of sentences is rather small in comparison to the variation of expressions. So having two articles that have multiple basic patterns in common is very unlikely. We extend the number of articles for obtaining basic patterns by using a cluster of comparable articles that report the same event instead of a single article. We call this cluster of articles a "basic cluster." Using basic clusters instead of single articles also helps to increase the redundancy of data. We can give more confidence to repeated basic patterns.

Note that the notion of "basic cluster" is different from the clusters used for creating tables explained above. In the following sections, a cluster for creating a table is called a "metacluster," because this is a cluster of basic clusters. A basic cluster consists of a set of articles that report the same event which happens at a certain time, and a metacluster consists of a set of events that contain the same relation over a certain period.

We try to increase the number of articles in a basic cluster by looking at multiple news sources simultaneously. We use a clustering algorithm that uses a vector-space-model to obtain basic clusters. Then we apply cross-document coreference resolution to connect entities of different articles within a basic cluster. This way, we can increase the number of basic patterns connected to each entity. Also, it allows us to give a weight to entities. We calculate their weights using the number of occurrences within a cluster and their position within an article. These entities are used to obtain basic patterns later.

We also use a parser and tree normalizer to generate basic patterns. The format of basic patterns is crucial to performance. We think a basic pattern should be somewhat specific, since each pattern should capture an entity with some relevant context. But at the same time a basic pattern should be general enough to reduce data sparseness. We choose a predicate-argument structure as a natural solution for this problem. Compared to traditional constituent trees, a predicate-argument structure is a higher-level representation of sentences that has gained wide acceptance from the natural language community recently. In this paper we used a logical feature structure called GLARF proposed by Meyers et al. (2001a). A GLARF converter takes a syntactic tree as an input and augments it with several

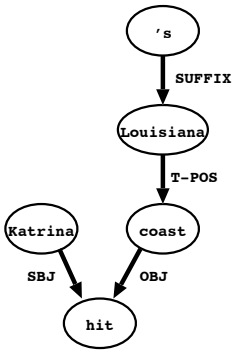


Figure 3: GLARF structure of the sentence "*Katrina hit Louisiana's coast.*"

features. Figure 3 shows a sample GLARF structure obtained from the sentence "*Katrina hit Louisiana's coast.*" We used GLARF for two reasons: first, unlike traditional constituent parsers, GLARF has an ability to regularize several linguistic phenomena such as participial constructions and coordination. This allows us to handle this syntactic variety in a uniform way. Second, an output structure can be easily converted into a directed graph that represents the relationship between each word, without losing significant information from the original sentence. Compared to an ordinary constituent tree, it is easier to extract syntactic relationships. In the next section, we discuss how we used this structure to generate basic patterns.

3 Implementation

The overall process to generate basic patterns and discover relations from unannotated news articles is shown in Figure 4. Theoretically this could be a straight pipeline, but due to the nature of the implementation we process some stages separately and combine them in the later stage. In the following subsection, we explain each component.

3.1 Web Crawling and Basic Clustering

First of all, we need a lot of news articles from multiple news sources. We created a simple web crawler that extract the main texts from web pages. We observed that the crawler can correctly take the main texts from about 90% of the pages from each news site. We ran the crawler every day on several news sites. Then we applied a simple clustering algorithm to the obtained articles in order to find a set of articles that talk about exactly the same news and form a basic cluster.

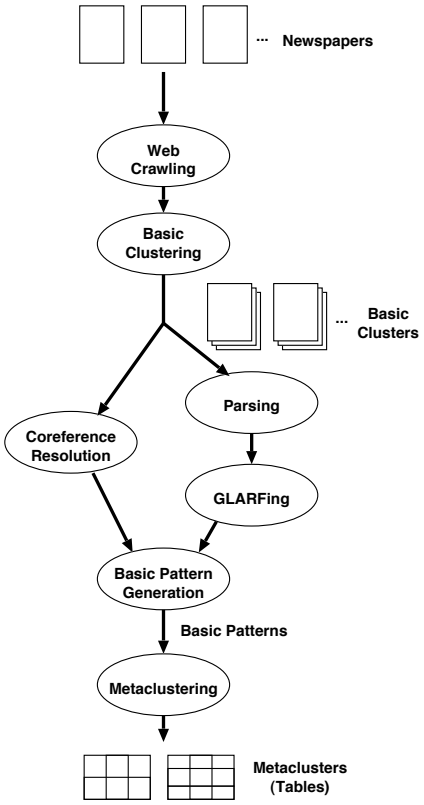


Figure 4: System overview.

We eliminate stop words and stem all the other words, then compute the similarity between two articles by using a bag-of-words approach. In news articles, a sentence that appears in the beginning of an article is usually more important than the others. So we preserved the word order to take into account the location of each sentence. First we computed a word vector from each article:

$$V_w(A) = \text{IDF}(w) \sum_{i \in POS(w,A)} \exp(-\frac{i}{avgwords})$$

where $V_w(A)$ is a vector element of word w in article A, $IDF(w)$ is the inverse document frequency of word w, and $POS(w, A)$ is a list of w's positions in the article. $avgwords$ is the average number of words for all articles. Then we calculated the cosine value of each pair of vectors:

$$Sim(A_1, A_2) = \cos(V(A_1) \cdot V(A_2))$$

We computed the similarity of all possible pairs of articles from the same day, and selected the pairs

whose similarity exceeded a certain threshold (0.65 in this experiment) to form a basic cluster.

3.2 Parsing and GLARFing

After getting a set of basic clusters, we pass them to an existing statistical parser (Charniak, 2000) and rule-based tree normalizer to obtain a GLARF structure for each sentence in every article. The current implementation of a GLARF converter gives about 75% F-score using parser output. For the details of GLARF representation and its conversion, see Meyers et al. (2001b).

3.3 NE Tagging and Coreference Resolution

In parallel with parsing and GLARFing, we also apply NE tagging and coreference resolution for each article in a basic cluster. We used an HMM-based NE tagger whose performance is about 85% in F-score. This NE tagger produces ACE-type Named Entities [1]: PERSON, ORGANIZATION, GPE, LOCATION and FACILITY [2]. After applying single-document coreference resolution for each article, we connect the entities among different articles in the same basic cluster to obtain cross-document coreference entities with simple string matching.

3.4 Basic Pattern Generation

After getting a GLARF structure for each sentence and a set of documents whose entities are tagged and connected to each other, we merge the two outputs and create a big network of GLARF structures whose nodes are interconnected across different sentences/articles. Now we can generate basic patterns for each entity. First, we compute the weight for each cross-document entity E in a certain basic cluster as follows:

$$W_E = \sum_{e \in E} mentions(e) \cdot \exp(-C \cdot firstsent(e))$$

where $e \in E$ is an entity within one article and $mentions(e)$ and $firstsent(e)$ are the number of mentions of entity e in a document and the position of the sentence where entity e first appeared, respectively. C is a constant value which was 0.5 in this experiment. To reduce combinatorial complexity, we took only the five most highly weighted entities from each basic cluster to generate basic patterns. We observed these five entities can cover major relations that are reported in a basic cluster.

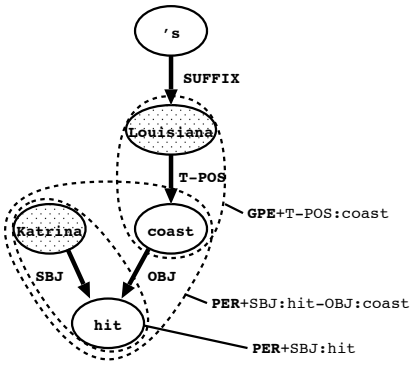


Figure 5: Basic patterns obtained from the sentence "*Katrina hit Louisiana's coast.*"

Next, we obtain basic patterns from the GLARF structures. We used only the first ten sentences in each article for getting basic patterns, as most important facts are usually written in the first few sentences of a news article. Figure 5 shows all the basic patterns obtained from the sentence "*Katrina hit Louisiana's coast.*" The shaded nodes "Katrina" and "Louisiana" are entities from which each basic pattern originates. We take a path of GLARF nodes from each entity node until it reaches any predicative node: noun, verb, or adjective in this case. Since the nodes "hit" and "coast" can be predicates in this example, we obtain three unique paths "*Louisiana*+T-POS:coast (*Louisiana's coast*)", "*Katrina*+SBJ:hit (*Katrina hit something*)", and "*Katrina*+SBJ:hit-OBJ:coast (*Katrina hit some coast*)".

To increase the specificity of patterns, we generate extra basic patterns by adding a node that is immediately connected to a predicative node. (From this example, we generate two basic patterns: "hit" and "hit-coast" from the "Katrina" node.)

Notice that in a GLARF structure, the type of each argument such as subject or object is preserved in an edge even if we extract a single path of a graph. Now, we replace both entities "Katrina" and "Louisiana" with variables

[1] The ACE task description can be found at http://www.itl.nist.gov/iad/894.01/tests/ace/ and the ACE guidelines at http://www.ldc.upenn.edu/Projects/ACE/

[2] The hurricane names used in the examples were recognized as PERSON.

based on their NE tags and obtain parameterized patterns: "*GPE*+T-POS:coast (*Louisiana's coast*)", "*PER*+SBJ:hit (*Katrina hit something*)", and "*PER*+SBJ:hit-OBJ:coast (*Katrina hit some coast*)".

After taking all the basic patterns from every basic cluster, we compute the Inverse Cluster Frequency (ICF) of each unique basic pattern. ICF is similar to the Inverse Document Frequency (IDF) of words, which is used to calculate the weight of each basic pattern for metaclustering.

3.5 Metaclustering

Finally, we can perform metaclustering to obtain tables. We compute the similarity between each basic cluster pair, as seen in Figure 6. X_A and X_B are the set of cross-document entities from basic clusters c_A and c_B, respectively. We examine all possible mappings of relations (parallel mappings of multiple entities) from both basic clusters, and find all the mappings M whose similarity score exceeds a certain threshold. wordsim(c_A, c_B) is the bag-of-words similarity of two clusters. As a weighting function we used ICF:

$$\text{weight}(p) = -\log(\frac{\text{clusters that include p}}{\text{all clusters}})$$

We then sort the similarities of all possible pairs of basic clusters, and try to build a metacluster by taking the most strongly connected pair first. Note that in this process we may assign one basic cluster to several different metaclusters. When a link is found between two basic clusters that were already assigned to a metacluster, we try to put them into all the existing metaclusters it belongs to. However, we allow a basic cluster to be added only if it can fill all the columns in that table. In other words, the first two basic clusters (i.e. an initial two-row table) determines its columns and therefore define the relation of that table.

4 Experiment and Evaluation

We used twelve newspapers published mainly in the U.S. We collected their articles over two months (from Sep. 21, 2005 - Nov. 27, 2005). We obtained 643,767 basic patterns and 7,990 unique types. Then we applied metaclustering to these basic clusters and obtained 302 metaclusters (tables). We then removed duplicated rows and took only the tables that had 3 or more rows. Finally we had 101 tables. The total number the of articles and clusters we used are shown in Table 2.

Source articles	28,009
Basic clusters	5,543
Basic patterns (token)	643,767
Basic patterns (type)	7,990
Metaclusters	302
Metaclusters (rows $\geq$ 3)	101

Table 2: Articles and obtained metaclusters.

4.1 Evaluation Method

We evaluated the obtained tables as follows. For each row in a table, we added a summary of the source articles that were used to extract the relation. Then for each table, an evaluator looks into every row and its source article, and tries to come up with a sentence that explains the relation among its columns. The description should be as specific as possible. If at least half of the rows can fit the explanation, the table is considered "consistent." For each consistent table, the evaluator wrote down the sentence using variable names ($1, $2, ...) to refer to its columns. Finally, we counted the number of consistent tables. We also counted how many rows in each table can fit the explanation.

4.2 Results

We evaluated 48 randomly chosen tables. Among these tables, we found that 36 tables were consistent. We also counted the total number of rows that fit each description, shown in Table 3. Table 4 shows the descriptions of the selected tables. The largest consistent table was about hurricanes (Table 5). Although we cannot exactly measure the recall of each table, we tried to estimate the recall by comparing this hurricane table to a manually created one (Table 6). We found 6 out of 9 hurricanes [3]. It is worth noting that most of these hurricane names were automatically disambiguated although our NE tagger didn't distinguish a hurricane name from a person

[3]Hurricane Katrina and Longwang shown in the previous examples are not included in this table. They appeared before this period.

```
for each cluster pair (c_A, c_B) {
    X_A = c_A.entities
    X_B = c_B.entities
    for each entity mapping M = [(x_A1, x_B1), ..., (x_An, x_Bn)] ∈ (2^|X_A| × 2^|X_B|) {
        for each entity pair (x_Ai, x_Bi) {
            P_i = x_Ai.patterns ∩ x_Bi.patterns
            pairscore_i = Σ_{p∈P_i} weight(p)
        }
        mapscore = Σ pairscore_i
        if T_1 < |M| and T_2 < mapscore and T_3 < wordsim(c_A.words, c_B.words) {
            link c_A and c_B with mapping M.
        }
    }
}
```

Figure 6: Computing similarity of basic clusters.

Tables:

Consistent tables	36 (75%)
Inconsistent tables	12
Total	48

Rows:

Rows that fit the description	118 (73%)
Rows not fitted	43
Total	161

Table 3: Evaluation results.

Description	Rows
Storm $1(PER) probably affected $2(GPE).	8/16
Nominee $2(PER) must be confirmed by $1(ORG).	4/7
$1(PER) urges $2(GPE) to make changes.	4/6
$1(GPE) launched an attack in $2(GPE).	3/5
$1(PER) ran against $2(PER) in an election.	4/5
$2(PER) visited $1(GPE) on a diplomatic mission.	2/4
$2(PER) beat $1(PER) in golf.	4/4
$2(GPE) soldier(s) were killed in $1(GPE).	3/3
$2(PER) ran for governor of $1(GPE).	2/3
Boxer $1(PER) fought boxer $2(PER).	3/3

Table 4: Description of obtained tables and the number of fitted/total rows.

name. The second largest table (about nominations of officials) is shown in Table 7.

We reviewed 10 incorrect rows from various tables and found 4 of them were due to coreference errors and one error was due to a parse error. The other 4 errors were due to multiple basic patterns distant from each other that happened to refer to a different event reported in the same cluster. The causes of the one remaining error was obscure. Most inconsistent tables were a mixture of multiple relations and some of their rows still looked consistent.

We have a couple of open questions. First, the overall recall of our system might be lower than existing IE systems, as we are relying on a cluster of comparable articles rather than a single document to discover an event. We might be able to improve this in the future by adjusting the basic clustering algorithm or weighting schema of basic patterns. Secondly, some combinations of basic patterns looked inherently vague. For example, we used the two basic patterns "`pitched`" and "`' s-series`" in the following sentence (the patterns are underlined):

> Ervin Santana pitched 5 1-3 gutsy innings in his postseason debut for the Angels, Adam Kennedy hit a go-ahead triple that sent Yankees outfielders crashing to the ground, and Los Angeles beat New York 5-3 Monday night in the decisive Game 5 of their AL playoff series.

It is not clear whether this set of patterns can yield any meaningful relation. We are not sure how much this sort of table can affect overall IE performance.

5 Conclusion

In this paper we proposed Preemptive Information Extraction as a new direction of IE research. As its key technique, we presented Unrestricted Relation Discovery that tries to find parallel correspondences between multiple entities in a document, and perform clustering using basic patterns as features. To increase the number of basic patterns, we used a cluster of comparable articles instead of a single document. We presented the implementation of our preliminary system and its outputs. We obtained dozens of usable tables.

Article	1:dump	2:coast
2005-09-21 (1)	Rita	Texas
2005-09-23	Rita	New Orleans
2005-09-25	Bush	Texas
2005-09-26	Damrey	Hainan
2005-09-27 (2)	Damrey	Vietnam
2005-10-01	Rita	Louisiana
2005-10-02	Otis	Mexico
2005-10-04	Longwang	China
2005-10-05	Stan	Mexico
2005-10-06	Tammy	Florida
2005-10-07	Tammy	Georgia
2005-10-19 (3)	Wilma	Florida
2005-10-25	Wilma	Cuba
2005-10-25	Wilma	Massachusetts
2005-10-28	Beta	Nicaragua
2005-11-20	Gamma	Florida

1. More than 2,000 National Guard troops were put on active-duty alert to assist as **Rita** slammed into the string of islands and headed west, perhaps toward **Texas**. ...
2. **Typhoon Damrey** smashed into **Vietnam** on Tuesday after killing nine people in China, ...
3. Oil markets have been watching **Wilma**'s progress nervously, ... but the threat to energy interests appears to have eased as forecasters predict **the storm** will turn toward **Florida**. ...

Table 5: Hurricane table ("Storm $1(PER) probably affected $2(GPE).") and the actual expressions we used for extraction.

Hurricane	Date (Affected Place)	Articles
Philippe	Sep 17-20 (?)	6
* Rita	Sep 17-26 (Louisiana, Texas, etc.)	566
* Stan	Oct 1-5 (Mexico, Nicaragua, etc.)	83
* Tammy	Oct 5-? (Georgia, Alabama)	18
Vince	Oct 8-11 (Portugal, Spain)	12
* Wilma	Oct 15-25 (Cuba, Honduras, etc.)	368
Alpha	Oct 22-24 (Haiti, Dominican Rep.)	80
* Beta	Oct 26-31 (Nicaragua, Honduras)	55
* Gamma	Nov 13-20 (Belize, etc.)	36

Table 6: Hurricanes in North America between mid-Sep. and Nov. (from Wikipedia). Rows with a star (*) were actually extracted. The number of the source articles that contained a mention of the hurricane is shown in the right column.

Article	1:confirm	2:be-confirmed
2005-09-21	Senate	Roberts
2005-10-03	Supreme Court	Miers
2005-10-20	Senate	Bush
2005-10-26	Senate	Sauerbrey
2005-10-31	Senate	Mr. Alito
2005-11-04	Senate	Alito
2005-11-17	Fed	Bernanke

Table 7: Nomination table ("Nominee $2(PER) must be confirmed by $1(ORG).")

Acknowledgements

This research was supported by the National Science Foundation under Grant IIS-00325657. This paper does not necessarily reflect the position of the U.S. Government. We would like to thank Prof. Ralph Grishman who provided useful suggestions and discussions.

References

Eugene Agichtein and L. Gravano. 2000. Snowball: Extracting Relations from Large Plaintext Collections. In *Proceedings of the 5th ACM International Conference on Digital Libraries (DL-00)*.

Sergey Brin. 1998. Extracting Patterns and Relations from the World Wide Web. In *WebDB Workshop at EDBT '98*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL-2000*.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *Proceedings of the Annual Meeting of Association of Computational Linguistics (ACL-04)*.

Adam Meyers, Ralph Grishman, Michiko Kosaka, and Shubin Zhao. 2001a. Covering Treebanks with GLARF. In *ACL/EACL Workshop on Sharing Tools and Resources for Research and Education*.

Adam Meyers, Michiko Kosaka, Satoshi Sekine, Ralph Grishman, and Shubin Zhao. 2001b. Parsing and GLARFing. In *Proceedings of RANLP-2001*, Tzigov Chark, Bulgaria.

Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ellen Riloff. 1996. Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96)*.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An Improved Extraction Pattern Representation Model for Automatic IE Pattern Acquisition. In *Proceedings of the Annual Meeting of Association of Computational Linguistics (ACL-03)*.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Unsupervised Discovery of Scenario-Level Patterns for Information Extraction. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-00)*.

Probabilistic Context-Free Grammar Induction Based on Structural Zeros

Mehryar Mohri
Courant Institute of Mathematical Sciences
and Google Research
251 Mercer Street
New York, NY 10012
mohri@cs.nyu.edu

Brian Roark
Center for Spoken Language Understanding
OGI at Oregon Health & Science University
20000 NW Walker Road
Beaverton, Oregon 97006
roark@cslu.ogi.edu

Abstract

We present a method for induction of concise and accurate probabilistic context-free grammars for efficient use in early stages of a multi-stage parsing technique. The method is based on the use of statistical tests to determine if a non-terminal combination is unobserved due to sparse data or hard syntactic constraints. Experimental results show that, using this method, high accuracies can be achieved with a non-terminal set that is orders of magnitude smaller than in typically induced probabilistic context-free grammars, leading to substantial speed-ups in parsing. The approach is further used in combination with an existing reranker to provide competitive WSJ parsing results.

1 Introduction

There is a very severe speed vs. accuracy tradeoff in stochastic context-free parsing, which can be explained by the grammar factor in the running-time complexity of standard parsing algorithms such as the CYK algorithm (Kasami, 1965; Younger, 1967). That algorithm has complexity $O(n^3|P|)$, where n is the length in words of the sentence parsed, and $|P|$ is the number of grammar productions. Grammar non-terminals can be split to encode richer dependencies in a stochastic model and improve parsing accuracy. For example, the parent of the left-hand side (LHS) can be annotated onto the label of the LHS category (Johnson, 1998), hence differentiating, for instance, between expansions of a VP with parent S and parent VP. Such annotations, however, tend to substantially increase the number of grammar productions as well as the ambiguity of the grammar, thereby significantly slowing down the parsing algorithm. In the case of bilexical grammars, where categories in binary grammars are annotated with their lexical heads, the grammar factor contributes an additional $O(n^2|V_D|^3)$ complexity, leading to an overall $O(n^5|V_D|^3)$ parsing complexity, where $|V_D|$ is the number of delexicalized non-terminals (Eisner, 1997). Even with special modifications to the basic CYK algorithm, such as those presented by Eisner and Satta (1999), improvements to the stochastic model are obtained at the expense of efficiency.

In addition to the significant cost in efficiency, increasing the non-terminal set impacts parameter estimation for the stochastic model. With more productions, much fewer observations per production are available and one is left with the hope that a subsequent smoothing technique can effectively deal with this problem, regardless of the number of non-terminals created. Klein and Manning (2003b) showed that, by making certain linguistically-motivated node label annotations, but avoiding certain other kinds of state splits (mainly lexical annotations) models of relatively high accuracy can be built without resorting to smoothing. The resulting grammars were small enough to allow for exhaustive CYK parsing; even so, parsing speed was significantly impacted by the state splits: the test-set parsing time reported was about 3s for average length sentences, with a memory usage of 1GB.

This paper presents an automatic method for deciding which state to split in order to create concise and accurate unsmoothed probabilistic context-free grammars (PCFGs) for *efficient* use in early stages of a multi-stage parsing technique. The method is based on the use of statistical tests to determine if a non-terminal combination is unobserved due to the limited size of the sample (*sampling zero*) or because it is grammatically impossible (*structural zero*). This helps introduce a relatively small number of new non-terminals with little additional parsing

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 312–319,
New York, June 2006.

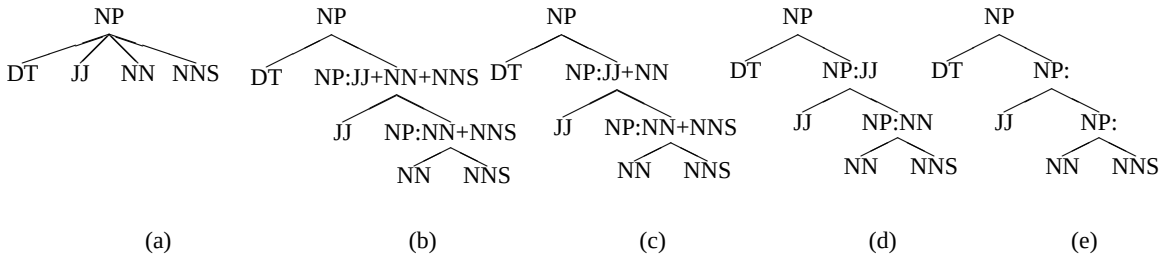


Figure 1: Five representations of an n-ary production, $n = 4$. (a) Original production (b) Right-factored production (c) Right-factored Markov order-2 (d) Right-factored Markov order-1 (e) Right-factored Markov order-0

overhead. Experimental results show that, using this method, high accuracies can be achieved with orders of magnitude fewer non-terminals than in typically induced PCFGs, leading to substantial speed-ups in parsing. The approach can further be used in combination with an existing reranker to provide competitive WSJ parsing results.

The remainder of the paper is structured as follows. Section 2 gives a brief description of PCFG induction from treebanks, including non-terminal label-splitting, factorization, and relative frequency estimation. Section 3 discusses the statistical criteria that we explored to determine structural zeros and thus select non-terminals for the factored PCFG. Finally, Section 4 reports the results of parsing experiments using our exhaustive k-best CYK parser with the concise PCFGs induced from the Penn WSJ treebank (Marcus et al., 1993).

2 Grammar induction

A context-free grammar $G = (V, T, S^{\dagger}, P)$, or CFG in short, consists of a set of non-terminal symbols V, a set of terminal symbols T, a start symbol $S^{\dagger} \in V$, and a set of production P of the form: $A \rightarrow \alpha$, where $A \in V$ and $\alpha \in (V \cup T)^*$. A PCFG is a CFG with a probability assigned to each production. Thus, the probabilities of the productions expanding a given non-terminal sum to one.

2.1 Smoothing and factorization

PCFGs induced from the Penn Treebank have many productions with long sequences of non-terminals on the RHS. Probability estimates of the RHS given the LHS are often smoothed by making a Markov assumption regarding the conditional independence of a category on those more than k categories away (Collins, 1997; Charniak, 2000):

$$\begin{aligned} \mathrm{P}(X \rightarrow Y_1 ... Y_n) &= \mathrm{P}(Y_1|X) \prod_{i=2}^{n} \mathrm{P}(Y_i|X, Y_1 \cdots Y_{i-1}) \\ &\approx \mathrm{P}(Y_1|X) \prod_{i=2}^{n} \mathrm{P}(Y_i|X, Y_{i-k} \cdots Y_{i-1}). \end{aligned}$$

Making such a Markov assumption is closely related to grammar transformations required for certain efficient parsing algorithms. For example, the CYK parsing algorithm takes as input a Chomsky Normal Form PCFG, i.e., a grammar where all productions are of the form $X \rightarrow Y Z$ or $X \rightarrow a$, where X, Y, and Z are non-terminals and a a terminal symbol.[1]. Binarized PCFGs are induced from a treebank whose trees have been factored so that n-ary productions with $n>2$ become sequences of $n-1$ binary productions. Full right-factorization involves concatenating the final $n-1$ categories from the RHS of an n-ary production to form a new composite non-terminal. For example, the original production NP → DT JJ NN NNS shown in Figure 1(a) is factored into three binary rules, as shown in Figure 1(b). Note that a PCFG induced from such right-factored trees is weakly equivalent to a PCFG induced from the original treebank, i.e., it describes the same language.

From such a factorization, one can make a Markov assumption for estimating the production probabilities by simply recording only the labels of the first k children dominated by the composite factored label. Figure 1 (c), (d), and (e) show right-factored trees of Markov orders 2, 1 and 0 respectively.[2] In addition to being used for smoothing

[1]Our implementation of the CYK algorithm has been extended to allow for unary productions with non-terminals on the RHS in the PCFG.

[2]Note that these factorizations do not provide exactly the stated Markov order for all dependencies in the productions, because we are restricting factorization to only produce binary productions. For example, in Figure 1(e), the probability of the

PCFG	Time (s)	Words/s	$\|V\|$	$\|P\|$	LR	LP	F
Right-factored	4848	6.7	10105	23220	69.2	73.8	71.5
Right-factored, Markov order-2	1302	24.9	2492	11659	68.8	73.8	71.3
Right-factored, Markov order-1	445	72.7	564	6354	68.0	73.0	70.5
Right-factored, Markov order-0	206	157.1	99	3803	61.2	65.5	63.3
Parent-annotated, Right-factored, Markov order-2	7510	4.3	5876	22444	76.2	78.3	77.2

Table 1: Baseline results of exhaustive CYK parsing using different probabilistic context-free grammars. Grammars are trained from sections 2-21 of the Penn WSJ Treebank and tested on all sentences of section 24 (no length limit), given weighted k-best POS-tagger output. The second and third columns report the total parsing time in seconds and the number of words parsed per second. The number of non-terminals, $|V|$, is indicated in the next column. The last three columns show the labeled recall (LR), labeled precision (LP), and F-measure (F).

as mentioned above, these factorizations reduce the size of the non-terminal set, which in turn improves CYK efficiency. The efficiency benefit of making a Markov assumption in factorization can be substantial, given the reduction of both non-terminals and productions, which improves the grammar constant. With standard right-factorization, as in Figure 1(b), the non-terminal set for the PCFG induced from sections 2-21 of the Penn WSJ Treebank grows from its original size of 72 to 10105, with 23220 productions. With a Markov factorization of orders 2, 1 and 0 we get non-terminal sets of size 2492, 564, and 99, and rule production sets of 11659, 6354, and 3803, respectively.

These reductions in the size of the non-terminal set from the original factored grammar result in an order of magnitude reduction in complexity of the CYK algorithm. One common strategy in statistical parsing is what can be termed an approximate coarse-to-fine approach: a simple PCFG is used to prune the search space to which richer and more complex models are applied subsequently (Charniak, 2000; Charniak and Johnson, 2005). Producing a "coarse" chart as efficiently as possible is thus crucial (Charniak et al., 1998; Blaheta and Charniak, 1999), making these factorizations particularly useful.

2.2 CYK parser and baselines

To illustrate the importance of this reduction in non-terminals for efficient parsing, we will present baseline parsing results for a development set. For these baseline trials, we trained a PCFG on sections 2-21 of the Penn WSJ Treebank (40k sentences, 936k words), and evaluated on section 24 (1346 sentences, 32k words). The parser takes as input the weighted k-best POS-tag sequences of a perceptron-trained tagger, using the tagger documented in Hollingshead et al. (2005). The number of tagger candidates k for all trials reported in this paper was $0.2n$, where n is the length of the string. From the weighted k-best list, we derive a conditional probability of each tag at position i by taking the sum of the exponential of the weights of all candidates with that tag at position i (softmax).

The parser is an exhaustive CYK parser that takes advantage of the fact that, with the grammar factorization method described, factored non-terminals can only occur as the second child of a binary production. Since the bulk of the non-terminals result from factorization, this greatly reduces the number of possible combinations given any two cells. When parsing with a parent-annotated grammar, we use a version of the parser that also takes advantage of the partitioning of the non-terminal set, i.e., the fact that any given non-terminal has already its parent indicated in its label, precluding combination with any non-terminal that does not have the same parent annotated.

Table 1 shows baseline results for standard right-factorization and factorization with Markov orders 0-2. Training consists of applying a particular grammar factorization to the treebank prior to inducing a PCFG using maximum likelihood (relative frequency) estimation. Testing consists of exhaustive CYK parsing of all sentences in the development set (no length limit) with the induced grammar, then detransforming the maximum likelihood parse back to the original format for evaluation against the reference parse. Evaluation includes the standard PARSEVAL measures labeled precision (LP) and labeled recall (LR), plus the harmonic mean (F-measure) of these two scores. We also present a result using parent annotation (Johnson, 1998) with a 2nd-order Markov assumption. Parent annotation occurs prior to treebank factorization. This condition is roughly equivalent to the $h = 1, v = 2$ in Klein and Manning

final NNS depends on the preceding NN, despite the Markov order-0 factorization. Because of our focus on efficient CYK, we accept these higher order dependencies rather than producing unary productions. Only n-ary rules n>2 are factored.

(2003b)[3].

From these results, we can see the large efficiency benefit of the Markov assumption, as the size of the non-terminal and production sets shrink. However, the efficiency gains come at a cost, with the Markov order-0 factored grammar resulting in a loss of a full 8 percentage points of F-measure accuracy. Parent annotation provides a significant accuracy improvement over the other baselines, but at a substantial efficiency cost.

Note that the efficiency impact is not a strict function of either the number of non-terminals or productions. Rather, it has to do with the number of competing non-terminals in cells of the chart. Some grammars may be very large, but less ambiguous in a way that reduces the number of cell entries, so that only a very small fraction of the productions need to be applied for any pair of cells. Parent annotation does just the opposite – it increases the number of cell entries for the same span, by creating entries for the same constituent with different parents. Some non-terminal annotations, e.g., splitting POS-tags by annotating their lexical items, result in a large grammar, but one where the number of productions that will apply for any pair of cells is greatly reduced.

Ideally, one would obtain the efficiency benefit of the small non-terminal set demonstrated with the Markov order-0 results, while encoding key grammatical constraints whose absence results in an accuracy loss. The method we present attempts to achieve this by using a statistical test to determine *structural zeros* and modifying the factorization to remove the probability mass assigned to them.

3 Detecting Structural Zeros

The main idea behind our method for detecting structural zeros is to search for events that are individually very frequent but that do not co-occur. For example, consider the Markov order-0 binary rule production in Figure 2. The production NP→NP NP: may be very frequent, as is the NP:→CC NN production, but they never co-occur together, because NP does not conjoin with NN in the Penn Treebank. If the counts of two such events a and b, e.g., NP→NP NP: and NP:→CC NN are very large, but the count of their co-occurrence is zero, then the co-occurrence of a and b can be viewed as a candidate for the list of events that are structurally inadmissible. The probability mass for the co-occurrence of a and b can be removed by replacing the factored non-terminal NP: with NP:CC:NN whenever there is a CC and an NN combining to form a factored NP non-terminal.

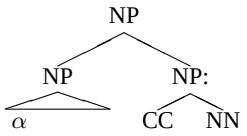


Figure 2: Markov order-0 local tree, with possible non-local ¡state-split information.

The expansion of the factored non-terminals is not the only event that we might consider. For example, a frequent left-most child of the first child of the production, or a common left-corner POS or lexical item, might never occur with certain productions. For example, 'SBAR→IN S' and 'IN→of' are both common productions, but they never co-occur. We focus on left-most children and left-corners because of the factorization that we have selected, but the same idea could be applied to other possible state splits.

Different statistical criteria can be used to compare the counts of two events with that of their co-occurrence. This section examines several possible criteria that are presented, for ease of exposition, with general sequences of events. For our specific purpose, these sequences of events would be two rule productions.

3.1 Notation

This section describes several statistical criteria to determine if a sequence of two events should be viewed as a structural zero. These tests can be generalized to longer and more complex sequences, and to various types of events, e.g., word, word class, or rule production sequences.

Given a corpus $\mathcal{C}$, and a vocabulary Σ, we denote by c_a the number of occurrences of a in $\mathcal{C}$. Let n be the total number of observations in $\mathcal{C}$. We will denote by $\bar{a}$ the set $\{b \in \Sigma : b \neq a\}$. Hence $c_{\bar{a}} = n - c_a$. Let $\mathrm{P}(a) = \frac{c_a}{n}$, and for $b \in \Sigma$, let $\mathrm{P}(a|b) = \frac{c_{ab}}{c_b}$. Note that $c_{\bar{a}b} = c_b - c_{ab}$.

[3]Their Markov order-2 factorization does not follow the linear order of the children, but rather includes the head-child plus one other, whereas our factorization does not involve identification of the head child.

3.2 Mutual information

The mutual information between two random variables X and Y is defined as

$$I(X;Y) = \sum_{x,y} \mathrm{P}(x,y) \log \frac{\mathrm{P}(x,y)}{\mathrm{P}(x)\mathrm{P}(y)}. \quad (1)$$

For a particular event sequence of length two ab, this suggests the following statistic:

$$\begin{aligned} I(ab) &= \log \mathrm{P}(ab) - \log \mathrm{P}(a) - \log \mathrm{P}(b) \\ &= \log c_{ab} - \log c_a - \log c_b + \log n \end{aligned}$$

Unfortunately, for $c_{ab} = 0$, $I(ab)$ is not finite. If we assume, however, that all unobserved sequences are given some ϵ count, then when $c_{ab} = 0$,

$$I(ab) = K - \log c_a - \log c_b, \quad (2)$$

where K is a constant. Since we need these statistics only for ranking purposes, we can ignore the constant factor.

3.3 Log odds ratio

Another statistic that, like mutual information, is ill-defined with zeros, is the *log odds ratio*:

$$\log(\hat{\theta}) = \log c_{ab} + \log c_{\bar{a}\bar{b}} - \log c_{\bar{a}b} - \log c_{a\bar{b}}.$$

Here again, if $c_{ab} = 0$, $\log(\hat{\theta})$ is not finite. But, if we assign to all unobserved pairs a small count ϵ, when $c_{ab} = 0$, $c_{\bar{a}b} = c_b$, and the expression becomes

$$\log(\hat{\theta}) = K + \log c_{\bar{a}\bar{b}} - \log c_b - \log c_a. \quad (3)$$

3.4 Pearson chi-squared

For any $i, j \in \Sigma$, define $\hat{\mu}_{ij} = \frac{c_i c_j}{n}$. The Pearson chi-squared test of independence is then defined as follows:

$$\mathcal{X}^2 = \sum_{\substack{i \in \{a,\bar{a}\} \\ j \in \{b,\bar{b}\}}} \frac{(c_{ij} - \hat{\mu}_{ij})^2}{\hat{\mu}_{ij}} = \sum_{\substack{i \in \{a,\bar{a}\} \\ j \in \{b,\bar{b}\}}} \frac{(nc_{ij} - c_i c_j)^2}{nc_i c_j}.$$

In the case of interest for us, $c_{ab} = 0$ and the statistic simplifies to:

$$\mathcal{X}^2 = \frac{c_a c_b}{n} + \frac{c_a^2 c_b}{nc_{\bar{a}}} + \frac{c_a c_b^2}{nc_{\bar{b}}} + \frac{c_a^2 c_b^2}{nc_{\bar{a}} c_{\bar{b}}} = \frac{nc_a c_b}{c_{\bar{a}} c_{\bar{b}}}. \quad (4)$$

3.5 Log likelihood ratio

Pearson's chi-squared statistic assumes a normal or approximately normal distribution, but that assumption typically does not hold for the occurrences of rare events (Dunning, 1994). It is then preferable to use the likelihood ratio statistic which allows us to compare the null hypothesis, that $\mathrm{P}(b) = \mathrm{P}(b|a) = \mathrm{P}(b|\bar{a}) = \frac{c_b}{n}$, with the hypothesis that $\mathrm{P}(b|a) = \frac{c_{ab}}{c_a}$ and $\mathrm{P}(b|\bar{a}) = \frac{c_{\bar{a}b}}{c_{\bar{a}}}$. In words, the null hypothesis is that the context of event a does not change the probability of seeing b. These discrete conditional probabilities follow a binomial distribution, hence the likelihood ratio is

$$\lambda = \frac{B[\mathrm{P}(b), c_{ab}, c_a]\ B[\mathrm{P}(b), c_{\bar{a}b}, c_{\bar{a}}]}{B[\mathrm{P}(b|a), c_{ab}, c_a]\ B[\mathrm{P}(b|\bar{a}), c_{\bar{a}b}, c_{\bar{a}}]}, \quad (5)$$

where $B[p, x, y] = p^x(1-p)^{y-x}\binom{y}{x}$. In the special case where $c_{ab} = 0$, $\mathrm{P}(b|\bar{a}) = \mathrm{P}(b)$, and this expression can be simplified as follows:

$$\begin{aligned} \lambda &= \frac{(1-\mathrm{P}(b))^{c_a}\mathrm{P}(b)^{c_{\bar{a}b}}(1-\mathrm{P}(b))^{c_{\bar{a}}-c_{\bar{a}b}}}{\mathrm{P}(b|\bar{a})^{c_{\bar{a}b}}(1-\mathrm{P}(b|\bar{a}))^{c_{\bar{a}}-c_{\bar{a}b}}} \\ &= (1-\mathrm{P}(b))^{c_a}. \end{aligned} \quad (6)$$

The log-likelihood ratio, denoted by G^2, is known to be asymptotically $\mathcal{X}^2$-distributed. In this case,

$$G^2 = -2c_a \log(1-\mathrm{P}(b)), \quad (7)$$

and with the binomial distribution, it has has one degree of freedom, thus the distribution will have asymptotically a mean of one and a standard deviation of $\sqrt{2}$.

We experimented with all of these statistics. While they measure different ratios, empirically they seem to produce very similar rankings. For the experiments reported in the next section, we used the log-likelihood ratio because this statistic is well-defined with zeros and is preferable to the Pearson chi-squared when dealing with rare events.

4 Experimental results

We used the log-likelihood ratio statistic G^2 to rank unobserved events ab, where $a \subset P$ and $b \in V$. Let V_o be the original, unfactored non-terminal set, and let $\alpha \in (V_o :)^*$ be a sequence of zero or more non-terminal/colon symbol pairs. Suppose we have a frequent factored non-terminal $X{:}\alpha B$ for $X, B \in V_o$. Then, if the set of productions $X \rightarrow YX{:}\alpha A$ with

$A \in V_o$ is also frequent, but $X \rightarrow YX{:}\alpha B$ is unobserved, this is a candidate structural zero. Similar splits can be considered with non-factored non-terminals.

There are two state split scenarios we consider in this paper. Scenario 1 is for factored non-terminals, which are always the second child of a binary production. For use in Equation 7,

$$\begin{aligned} c_a &= \sum_{A \in V_o} c(X \rightarrow YX{:}\alpha A) \\ c_b &= c(X{:}\alpha B) \quad \text{for } B \in V_o \\ c_{ab} &= c(X \rightarrow YX{:}\alpha B) \\ \mathrm{P}(b) &= \frac{c(X{:}\alpha B)}{\sum_{A \in V_o} c(X{:}\alpha A)}. \end{aligned}$$

Scenario 2 is for non-factored non-terminals, which we will split using the leftmost child, the left-corner POS-tag, and the left-corner lexical item, which are easily incorporated into our grammar factorization approach. In this scenario, the non-terminal to be split can be either the left or right child in the binary production. Here we show the counts for the left child case for use in Equation 7:

$$\begin{aligned} c_a &= \sum_{A} c(X \rightarrow Y[\alpha A]Z) \\ c_b &= c(\mathrm{Y}[\alpha B]) \\ c_{ab} &= c(X \rightarrow Y[\alpha B]Z) \\ \mathrm{P}(b) &= \frac{c(Y[\alpha B])}{\sum_{A} c(Y[\alpha A])} \end{aligned}$$

In this case, the possible splits are more complicated than just non-terminals as used in factoring. Here, the first possible split is the left child category, along with an indication of whether it is a unary production. One can further split by including the left-corner tag, and even further by including the left-corner word. For example, a unary S category might be split as follows: first to S[1:VP] if the single child of the S is a VP; next to S[1:VP:VBD] if the left-corner POS-tag is VBD; finally to S[1:VP:VBD:went] if the VBD verb was 'went'.

Note that, once non-terminals are split by annotating such information, the base non-terminals, e.g., S, implicitly encode contexts other than the ones that were split.

Table 2 shows the unobserved rules with the largest G^2 score, along with the ten non-terminals that these productions suggest for inclusion in our non-terminal set. The highest scoring unobserved production is PP → IN[that] NP. It receives such a high score because the base production (PP → IN NP) is very frequent, and so is 'IN→that', but they jointly never occur, since 'IN→that' is a complementizer. This split non-terminal also shows up in the second-highest ranked zero, an SBAR with 'that' complementizer and an S child that consists of a unary VP. The unary S→VP production is very common, but never with a 'that' complementizer in an SBAR.

Unobserved production (added NT(s) in bold)	G^2 score
PP → **IN[that]** NP	7153.1
SBAR → IN[that] **S[1:VP]**	5712.1
SBAR → **IN[of]** S	5270.5
SBAR → **WHNP[1:WDT]** **S[1:VP:TO]**	4299.9
VP → AUX **VP[MD]**	3972.1
SBAR → **IN[in]** S	3652.1
NP → NP **VP[VB]**	3236.2
NP → NN **NP:CC:NP**	2796.3
SBAR → WHNP **S[1:VP:VBG]**	2684.9

Table 2: Top ten non-terminals to add, and the unobserved productions leading to their addition to the non-terminal set.

Note that the fourth-ranked production uses two split non-terminals. The fifth ranked rule presumably does not add much information to aid parsing disambiguation, since the AUX MD tag sequence is unlikely[4]. The eighth ranked production is the first with a factored category, ruling out coordination between NN and NP.

Before presenting experimental results, we will mention some practical issues related to the approach described. First, we independently parameterized the number of factored categories to select and the number of non-factored categories to select. This was done to allow for finer control of the amount of splitting of non-terminals of each type. To choose 100 of each, every non-terminal was assigned the score of the highest scoring unobserved production within which it occurred. Then the 100 highest scoring non-terminals of each type were added to the base non-terminal list, which originally consisted of the atomic treebank non-terminals and Markov order-0 factored non-terminals.

Once the desired non-terminals are selected, the training corpus is factored, and non-terminals are split if they were among the selected set. Note, how-

[4]In fact, we do not consider splits when both siblings are POS-tags, because these are unlikely to carry any syntactic disambiguation.

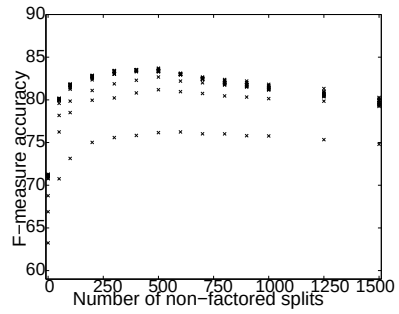


Figure 3: F-measure accuracy on development set versus the number of non-factored splits for the given run. Points represent different numbers of factored splits.

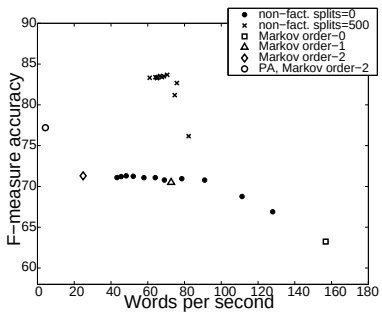


Figure 4: F-measure accuracy versus words-per-second for (1) no non-factored splits (i.e., only factored categories selected); (2) 500 non-factored splits, which was the best performing; and (3) four baseline results.

ever, that some of the information in a selected non-terminal may not be fully available, requiring some number of additional splits. Any non-terminal that is required by a selected non-terminal will be selected itself. For example, suppose that NP:CC:NP was chosen as a factored non-terminal. Then the second child of any local tree with that non-terminal on the LHS must either be an NP or a factored non-terminal with at least the first child identified as an NP, i.e., NP:NP. If that factored non-terminal was not selected to be in the set, it must be added. The same situation occurs with left-corner tags and words, which may be arbitrarily far below the category.

After factoring and selective splitting of non-terminals, the resulting treebank corpus is used to train a PCFG. Recall that we use the k-best output of a POS-tagger to parse. For each POS-tag and lexical item pair from the output of the tagger, we reduce the word to lower case and check to see if the combination is in the set of split POS-tags, in which case we split the tag, e.g., IN[that].

Figure 3 shows the F-measure accuracy for our trials on the development set versus the number of non-factored splits parameterized for the trial. From this plot, we can see that 500 non-factored splits provides the best F-measure accuracy on the dev set. Presumably, as more than 500 splits are made, sparse data becomes more problematic. Figure 4 shows the development set F-measure accuracy versus the number of words-per-second it takes to parse the development set, for non-factored splits of 0 and 500, at a range of factored split parameterizations. With 0 non-factored splits, efficiency is substantially impacted by increasing the factored splits, whereas it can be seen that with 500 non-factored splits, that impact is much less, so that the best performance is reached with both relatively few factored non-terminal splits, and a relatively small efficiency impact. The non-factored splits provide substantial accuracy improvements at relatively small efficiency cost.

Table 3 shows the 1-best and reranked 50-best results for the baseline Markov order-2 model, and the best-performing model using factored and non-factored non-terminal splits. We present the efficiency of the model in terms of words-per-second over the entire dev set, including the longer strings (maximum length 116 words)[5]. We used the k-best decoding algorithm of Huang and Chiang (2005) with our CYK parser, using on-demand k-best backpointer calculation. We then trained a MaxEnt reranker on sections 2-21, using the approach outlined in Charniak and Johnson (2005), via the publicly available reranking code from that paper.[6] We used the default features that come with that package. The processing time in the table includes the time to parse and rerank. As can be seen from the trials, there is some overhead to these processes, but the time is still dominated by the base parsing.

We present the k-best results to demonstrate the benefits of using a better model, such as the one we have presented, for producing candidates for downstream processing. Even with severe pruning to only the top 50 candidate parses per string, which results in low oracle and reranked accuracy for the Markov order-2 model, the best-performing model based on structural zeros achieves a relatively high oracle accuracy, and reaches 88.0 and 87.5 percent F-measure accuracy on the dev (f24) and eval (f23) sets respectively. Note that the well-known Char-

[5]The parsing time with our model for average length sentences (23-25 words) is 0.16 seconds per sentence.

[6]`http://www.cog.brown.edu/~mj/code`.

Technique	No. of Cands	Development (f24) Time(s)	Words/s	Oracle F	LR	LP	F	Eval (f23) LR	LP	F
Baseline, Markov order-2	1	1302	24.9	71.3	68.8	73.8	71.3	68.9	73.9	71.4
	50	1665	19.4	86.2	79.7	83.3	81.5	80.5	84.0	82.2
NT splits: factored=200	1	491	65.9	83.7	83.1	84.3	83.7	82.4	83.4	82.9
non-factored=500	50	628	51.5	93.8	87.4	88.7	88.0	87.1	88.0	87.5

Table 3: Parsing results on the development set (f24) and the evaluation set (f23) for the baseline Markov order-2 model and the best-performing structural zero model, with 200 factored and 500 non-factored non-terminal splits. 1-best results, plus reranking using a trained version of an existing reranker with 50 candidates.

niak parser (Charniak, 2000; Charniak and Johnson, 2005) uses a Markov order-3 baseline PCFG in the initial pass, with a best-first algorithm that is run past the first parse to populate the chart for use by the richer model. While we have demonstrated exhaustive parsing efficiency, our model could be used with any of the efficient search best-first approaches documented in the literature, from those used in the Charniak parser (Charniak et al., 1998; Blaheta and Charniak, 1999) to A* parsing (Klein and Manning, 2003a). By using a richer grammar of the sort we present, far fewer edges would be required in the chart to include sufficient quality candidates for the richer model, leading to further downstream savings of processing time.

5 Conclusion

We described a method for creating concise PCFGs by detecting structural zeros. The resulting unsmoothed PCFGs have far higher accuracy than simple induced PCFGs and yet are very efficient to use. While we focused on a small number of simple non-terminal splits that fit the factorization we had selected, the technique presented is applicable to a wider range of possible non-terminal annotations, including head or parent annotations. More generally, the ideas and method for determining structural zeros (vs. sampling zeros) can be used in other contexts for a variety of other learning tasks.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant IIS-0447214. 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. The first author's work was partially funded by the New York State Office of Science Technology and Academic Research (NYSTAR).

References

D. Blaheta and E. Charniak. 1999. Automatic compensation for parser figure-of-merit flaws. In *Proceedings of ACL*, pages 513–518.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of ACL*, pages 173–188.

E. Charniak, S. Goldwater, and M. Johnson. 1998. Edge-based best-first chart parsing. In *Proceedings of the 6th Workshop on Very Large Corpora*, pages 127–133.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*, pages 132–139.

M.J. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL*, pages 16–23.

T. Dunning. 1994. Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics*, 19(1):61–74.

J. Eisner and G. Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of ACL*, pages 457–464.

J. Eisner. 1997. Bilexical grammars and a cubic-time probabilistic parser. In *Proceedings of the International Workshop on Parsing Technologies*, pages 54–65.

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.

L. Huang and D. Chiang. 2005. Better k-best parsing. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*, pages 53–64.

M. Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):617–636.

T. Kasami. 1965. An efficient recognition and syntax analysis algorithm for context-free languages. Technical Report, AFCRL-65-758, Air Force Cambridge Research Lab., Bedford, MA.

D. Klein and C. Manning. 2003a. A* parsing: Fast exact Viterbi parse selection. In *Proceedings of HLT-NAACL*.

D. Klein and C. Manning. 2003b. Accurate unlexicalized parsing. In *Proceedings of ACL*.

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.

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

Prototype-Driven Learning for Sequence Models

Aria Haghighi
Computer Science Division
University of California Berkeley
aria42@cs.berkeley.edu

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

Abstract

We investigate *prototype-driven* learning for primarily unsupervised sequence modeling. Prior knowledge is specified declaratively, by providing a few canonical examples of each target annotation label. This sparse prototype information is then propagated across a corpus using distributional similarity features in a log-linear generative model. On part-of-speech induction in English and Chinese, as well as an information extraction task, prototype features provide substantial error rate reductions over competitive baselines and outperform previous work. For example, we can achieve an English part-of-speech tagging accuracy of 80.5% using only three examples of each tag and no dictionary constraints. We also compare to semi-supervised learning and discuss the system's error trends.

1 Introduction

Learning, broadly taken, involves choosing a good model from a large space of possible models. In supervised learning, model behavior is primarily determined by labeled examples, whose production requires a certain kind of expertise and, typically, a substantial commitment of resources. In unsupervised learning, model behavior is largely determined by the structure of the model. Designing models to exhibit a certain target behavior requires another, rare kind of expertise and effort. Unsupervised learning, while minimizing the usage of labeled data, does not necessarily minimize total effort. We therefore consider here how to learn models with the least effort. In particular, we argue for a certain kind of semi-supervised learning, which we call *prototype-driven* learning.

In prototype-driven learning, we specify prototypical examples for each target label or label configuration, but do not necessarily label any documents or sentences. For example, when learning a model for Penn treebank-style part-of-speech tagging in English, we may list the 45 target tags and a few examples of each tag (see figure 4 for a concrete prototype list for this task). This manner of specifying prior knowledge about the task has several advantages. First, is it certainly compact (though it remains to be proven that it is effective). Second, it is more or less the minimum one would have to provide to a human annotator in order to specify a new annotation task and policy (compare, for example, with the list in figure 2, which suggests an entirely different task). Indeed, prototype lists have been used pedagogically to summarize tagsets to students (Manning and Schütze, 1999). Finally, natural language does exhibit proform and prototype effects (Radford, 1988), which suggests that learning by analogy to prototypes may be effective for language tasks.

In this paper, we consider three sequence modeling tasks: part-of-speech tagging in English and Chinese and a classified ads information extraction task. Our general approach is to use distributional similarity to link any given word to similar prototypes. For example, the word *reported* may be linked to *said*, which is in turn a prototype for the part-of-speech VBD. We then encode these prototype links as features in a log-linear generative model, which is trained to fit unlabeled data (see section 4.1). Distributional prototype features provide substantial error rate reductions on all three tasks. For example, on English part-of-speech tagging with three prototypes per tag, adding prototype features to the baseline raises per-position accuracy from 41.3% to 80.5%.

2 Tasks and Related Work: Tagging

For our part-of-speech tagging experiments, we used data from the English and Chinese Penn treebanks (Marcus et al., 1994; Ircs, 2002). Example sentences

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 320–327,
New York, June 2006.

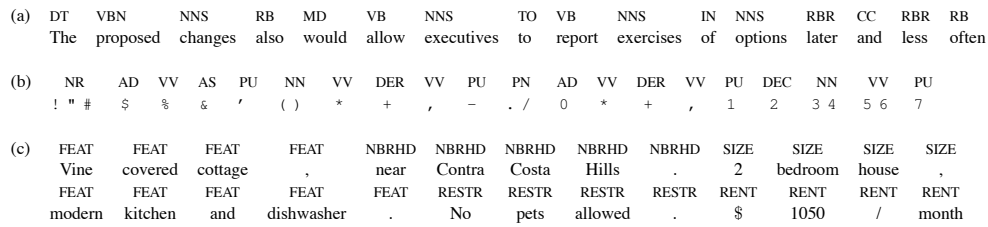


Figure 1: Sequence tasks: (a) English POS, (b) Chinese POS, and (c) Classified ad segmentation

are shown in figure 1(a) and (b). A great deal of research has investigated the unsupervised and semi-supervised induction of part-of-speech models, especially in English, and there is unfortunately only space to mention some highly related work here.

One approach to unsupervised learning of part-of-speech models is to induce HMMs from unlabeled data in a maximum-likelihood framework. For example, Merialdo (1991) presents experiments learning HMMs using EM. Merialdo's results most famously show that re-estimation degrades accuracy unless almost no examples are labeled. Less famously, his results also demonstrate that re-estimation can improve tagging accuracies to some degree in the fully unsupervised case.

One recent and much more successful approach to part-of-speech learning is *contrastive estimation*, presented in Smith and Eisner (2005). They utilize task-specific comparison neighborhoods for part-of-speech tagging to alter their objective function.

Both of these works require specification of the legal tags for each word. Such dictionaries are large and embody a great deal of lexical knowledge. A prototype list, in contrast, is extremely compact.

3 Tasks and Related Work: Extraction

Grenager et al. (2005) presents an unsupervised approach to an information extraction task, called CLASSIFIEDS here, which involves segmenting classified advertisements into topical sections (see figure 1(c)). Labels in this domain tend to be "sticky" in that the correct annotation tends to consist of multi-element fields of the same label. The overall approach of Grenager et al. (2005) typifies the process involved in fully unsupervised learning on new domain: they first alter the structure of their HMM so that diagonal transitions are preferred, then modify the transition structure to explicitly model boundary tokens, and so on. Given enough refinements the model learns to segment with a reasonable match to the target structure.

Label	Prototypes
ROOMATES	`roommate respectful drama`
RESTRICTIONS	`pets smoking dog`
UTILITIES	`utilities pays electricity`
AVAILABLE	`immediately begin cheaper`
SIZE	`2 br sq`
PHOTOS	`pictures image link`
RENT	`$ month *number*15*1`
CONTACT	`*phone* call *time*`
FEATURES	`kitchen laundry parking`
NEIGHBORHOOD	`close near shopping`
ADDRESS	`address carlmont *ordinal*5`
BOUNDARY	`; . !`

Figure 2: Prototype list derived from the development set of the CLASSIFIEDS data. The BOUNDARY field is not present in the original annotation, but added to model boundaries (see Section 5.3). The starred tokens are the results of collapsing of basic entities during pre-processing as is done in (Grenager et al., 2005)

In section 5.3, we discuss an approach to this task which does not require customization of model structure, but rather centers on feature engineering.

4 Approach

In the present work, we consider the problem of learning sequence models over text. For each document $x = [x_i]$, we would like to predict a sequence of labels $y = [y_i]$, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. We construct a generative model, $p(x, y|\theta)$, where θ are the model's parameters, and choose parameters to maximize the log-likelihood of our observed data D:

$$\begin{aligned} \mathcal{L}(\theta; D) &= \sum_{x \in D} \log p(x|\theta) \\ &= \sum_{x \in D} \log \sum_{y} p(x, y|\theta) \end{aligned}$$

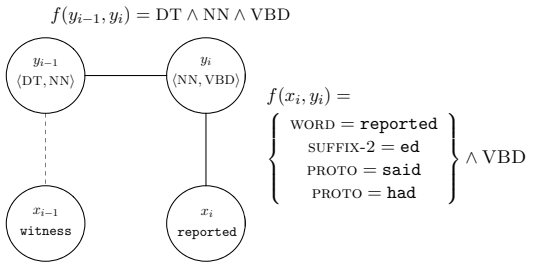


Figure 3: Graphical model representation of trigram tagger for English POS domain.

4.1 Markov Random Fields

We take our model family to be chain-structured Markov random fields (MRFs), the undirected equivalent of HMMs. Our joint probability model over (x, y) is given by

$$p(x, y|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \phi(x_i, y_i)\phi(y_{i-1}, y_i)$$

where $\phi(c)$ is a potential over a clique c, taking the form $\exp\left\{\theta^T f(c)\right\}$, and $f(c)$ is the vector of features active over c. In our sequence models, the cliques are over the edges/transitions (y_{i-1}, y_i) and nodes/emissions (x_i, y_i). See figure 3 for an example from the English POS tagging domain.

Note that the only way an MRF differs from a conditional random field (CRF) (Lafferty et al., 2001) is that the partition function is no longer observation dependent; we are modeling the joint probability of x and y instead of y given x. As a result, learning an MRF is slightly harder than learning a CRF; we discuss this issue in section 4.4.

4.2 Prototype-Driven Learning

We assume prior knowledge about the target structure via a *prototype list*, which specifies the set of target labels $\mathcal{Y}$ and, for each label $y \in \mathcal{Y}$, a set of prototypes words, $p_y \in \mathcal{P}_y$. See figures 2 and 4 for examples of prototype lists.[1]

[1]Note that this setting differs from the standard semi-supervised learning setup, where a small number of fully labeled examples are given and used in conjunction with a larger amount of unlabeled data. In our prototype-driven approach, we never provide a single *fully* labeled example sequence. See section 5.3 for further comparison of this setting to semi-supervised learning.

Broadly, we would like to learn sequence models which both explain the observed data and meet our prior expectations about target structure. A straightforward way to implement this is to constrain each prototype word to take only its given label(s) at training time. As we show in section 5, this does not work well in practice because this constraint on the model is very sparse.

In providing a prototype, however, we generally mean something stronger than a constraint on that word. In particular, we may intend that words which are in some sense similar to a prototype generally be given the same label(s) as that prototype.

4.3 Distributional Similarity

In syntactic distributional clustering, words are grouped on the basis of the vectors of their preceeding and following words (Schütze, 1995; Clark, 2001). The underlying linguistic idea is that replacing a word with another word of the same syntactic category should preserve syntactic well-formedness (Radford, 1988). We present more details in section 5, but for now assume that a similarity function over word types is given.

Suppose further that for each non-prototype word type w, we have a subset of prototypes, S_w, which are known to be distributionally similar to w (above some threshold). We would like our model to relate the tags of w to those of S_w.

One approach to enforcing the distributional assumption in a sequence model is by supplementing the training objective (here, data likelihood) with a penalty term that encourages parameters for which each w's posterior distribution over tags is compatible with it's prototypes S_w. For example, we might maximize,

$$\sum_{x \in D} \log p(x|\theta) - \sum_{w} \sum_{z \in S_w} KL(\, t|z \,||\, t|w)$$

where $t|w$ is the model's distribution of tags for word w. The disadvantage of a penalty-based approach is that it is difficult to construct the penalty term in a way which produces exactly the desired behavior.

Instead, we introduce distributional prototypes into the learning process as features in our log-linear model. Concretely, for each prototype z, we introduce a predicate PROTO $= z$ which becomes active

at each w for which $z \in S_w$ (see figure 3). One advantage of this approach is that it allows the strength of the distributional constraint to be calibrated along with any other features; it was also more successful in our experiments.

4.4 Parameter Estimation

So far we have ignored the issue of how we learn model parameters θ which maximize $\mathcal{L}(\theta; D)$. If our model family were HMMs, we could use the EM algorithm to perform a local search. Since we have a log-linear formulation, we instead use a gradient-based search. In particular, we use L-BFGS (Liu and Nocedal, 1989), a standard numerical optimization technique, which requires the ability to evaluate $\mathcal{L}(\theta; D)$ and its gradient at a given θ.

The density $p(x|\theta)$ is easily calculated up to the global constant $Z(\theta)$ using the forward-backward algorithm (Rabiner, 1989). The partition function is given by

$$\begin{aligned} Z(\theta) &= \sum_x \sum_y \prod_{i=1}^{n} \phi(x_i, y_i)\phi(y_{i-1}, y_i) \\ &= \sum_x \sum_y \mathit{score}(x, y) \end{aligned}$$

$Z(\theta)$ can be computed exactly under certain assumptions about the clique potentials, but can in all cases be bounded by

$$\hat{Z}(\theta) = \sum_{\ell=1}^{K} \hat{Z}_\ell(\theta) = \sum_{\ell=1}^{K} \sum_{x:|x|=\ell} \mathit{score}(x, y)$$

Where K is a suitably chosen large constant. We can efficiently compute $\hat{Z}_\ell(\theta)$ for fixed ℓ using a generalization of the forward-backward algorithm to the lattice of all observations x of length ℓ (see Smith and Eisner (2005) for an exposition).

Similar to supervised maximum entropy problems, the partial derivative of $\mathcal{L}(\theta; D)$ with respect to each parameter θ_j (associated with feature f_j) is given by a difference in feature expectations:

$$\frac{\partial \mathcal{L}(\theta; D)}{\partial \theta_j} = \sum_{x \in D} \left(\mathbb{E}_{y|x,\theta} f_j - \mathbb{E}_{x,y|\theta} f_j \right)$$

The first expectation is the expected count of the feature under the model's $p(y|x, \theta)$ and is again easily computed with the forward-backward algorithm, just as for CRFs or HMMs. The second expectation is the expectation of the feature under the model's joint distribution over all x, y pairs, and is harder to calculate. Again assuming that sentences beyond a certain length have negligible mass, we calculate the expectation of the feature for each fixed length ℓ and take a (truncated) weighted sum:

$$\mathbb{E}_{x,y|\theta} f_j = \sum_{\ell=1}^{K} p(|x| = \ell) \mathbb{E}_{x,y|\ell,\theta} f_j$$

For fixed ℓ, we can calculate $\mathbb{E}_{x,y|\ell,\theta} f_j$ using the lattice of all inputs of length ℓ. The quantity $p(|x| = \ell)$ is simply $\hat{Z}_\ell(\theta)/\hat{Z}(\theta)$.

As regularization, we use a diagonal Gaussian prior with variance $\sigma^2 = 0.5$, which gave relatively good performance on all tasks.

	Num Tokens	
Setting	48K	193K
BASE	42.2	41.3
PROTO	61.9	68.8
PROTO+SIM	79.1	80.5

Table 1: English POS results measured by per-position accuracy

5 Experiments

We experimented with prototype-driven learning in three domains: English and Chinese part-of-speech tagging and classified advertisement field segmentation. At inference time, we used maximum posterior decoding,[2] which we found to be uniformly but slightly superior to Viterbi decoding.

5.1 English POS Tagging

For our English part-of-speech tagging experiments, we used the WSJ portion of the English Penn treebank (Marcus et al., 1994). We took our data to be either the first 48K tokens (2000 sentences) or 193K tokens (8000 sentences) starting from section 2. We used a trigram tagger of the model form outlined in section 4.1 with the same set of spelling features reported in Smith and Eisner (2005): exact word type,

[2] At each position choosing the label which has the highest posterior probability, obtained from the forward-backward algorithm.

Label	Prototype	Label	Prototype
NN	% company year	NNS	years shares companies
JJ	new other last	VBG	including being according
MD	will would could	-LRB-	-LRB- -LCB-
VBP	are 're 've	DT	the a The
RB	n't also not	WP$	whose
-RRB-	-RRB- -RCB-	FW	bono del kanji
WRB	when how where	RP	Up ON
IN	of in for	VBD	said was had
SYM	c b f	$	$ US$ C$
CD	million billion two	#	#
TO	to To na	:	– : ;
VBN	been based compared	NNPS	Philippines Angels Rights
RBR	Earlier duller	“	“ ‘ non-“
VBZ	is has says	VB	be take provide
JJS	least largest biggest	RBS	Worst
NNP	Mr. U.S. Corp.	,	,
POS	'S	CC	and or But
PRP$	its their his	JJR	smaller greater larger
PDT	Quite	WP	who what What
WDT	which Whatever whatever	.	. ? !
EX	There	PRP	it he they
”	”	UH	Oh Well Yeah

Figure 4: English POS prototype list

Correct Tag	Predicted Tag	% of Errors
CD	DT	6.2
NN	JJ	5.3
JJ	NN	5.2
VBD	VBN	3.3
NNS	NN	3.2

Figure 5: Most common English POS confusions for PROTO+SIM on 193K tokens

character suffixes of length up to 3, *initial-capital*, *contains-hyphen*, and *contains-digit*. Our only edge features were tag trigrams.

With just these features (our baseline BASE) the problem is symmetric in the 45 model labels. In order to break initial symmetry we initialized our potentials to be near one, with some random noise. To evaluate in this setting, model labels must be mapped to target labels. We followed the common approach in the literature, greedily mapping each model label to a target label in order to maximize per-position accuracy on the dataset. The results of BASE, reported in table 1, depend upon random initialization; averaging over 10 runs gave an average per-position accuracy of 41.3% on the larger training set.

We automatically extracted the prototype list by taking our data and selecting for each annotated label the top three occurring word types which were not given another label more often. This resulted in 116 prototypes for the 193K token setting.[3] For comparison, there are 18,423 word types occurring in this data.

Incorporating the prototype list in the simplest possible way, we fixed prototype occurrences in the data to their respective annotation labels. In this case, the model is no longer symmetric, and we no longer require random initialization or post-hoc mapping of labels. Adding prototypes in this way gave an accuracy of 68.8% on all tokens, but only 47.7% on non-prototype occurrences, which is only a marginal improvement over BASE. It appears as though the prototype information is not spreading to non-prototype words.

In order to remedy this, we incorporated distributional similarity features. Similar to (Schütze, 1995), we collect for each word type a context vector of the counts of the most frequent 500 words, conjoined with a direction and distance (e.g +1,-2). We then performed an SVD on the matrix to obtain a reduced rank approximation. We used the dot product between left singular vectors as a measure of distributional similarity. For each word w, we find the set of prototype words with similarity exceeding a fixed threshold of 0.35. For each of these prototypes z, we add a predicate PROTO $= z$ to each occurrence of w. For example, we might add PROTO $=$ *said* to each token of *reported* (as in figure 3).[4]

Each prototype word is also its own prototype (since a word has maximum similarity to itself), so when we lock the prototype to a label, we are also pushing all the words distributionally similar to that prototype towards that label.[5]

[3]To be clear: this method of constructing a prototype list required statistics from the labeled data. However, we believe it to be a fair and necessary approach for several reasons. First, we wanted our results to be repeatable. Second, we did not want to overly tune this list, though experiments below suggest that tuning could greatly reduce the error rate. Finally, it allowed us to run on Chinese, where the authors have no expertise.

[4]Details of distributional similarity features: To extract context vectors, we used a window of size 2 in either direction and use the first 250 singular vectors. We collected counts from all the WSJ portion of the Penn Treebank as well as the entire BLIPP corpus. We limited each word to have similarity features for its top 5 most similar prototypes.

[5]Note that the presence of a prototype feature does not ensure every instance of that word type will be given its prototype's label; pressure from "edge" features or other prototype features can cause occurrences of a word type to be given different labels. However, rare words with a single prototype feature are almost always given that prototype's label.

This setting, PROTO+SIM, brings the all-tokens accuracy up to 80.5%, which is a 37.5% error reduction over PROTO. For non-prototypes, the accuracy increases to 67.8%, an error reduction of 38.4% over PROTO. The overall error reduction from BASE to PROTO+SIM on all-token accuracy is 66.7%.

Table 5 lists the most common confusions for PROTO+SIM. The second, third, and fourth most common confusions are characteristic of fully supervised taggers (though greater in number here) and are difficult. For instance, both JJs and NNs tend to occur after determiners and before nouns. The CD and DT confusion is a result of our prototype list not containing a *contains-digit* prototype for CD, so the predicate fails to be linked to CDs. Of course in a realistic, iterative design setting, we could have altered the prototype list to include a *contains-digit* prototype for CD and corrected this confusion.

Figure 6 shows the marginal posterior distribution over label pairs (roughly, the bigram transition matrix) according to the treebank labels and the PROTO+SIM model run over the training set (using a collapsed tag set for space). Note that the broad structure is recovered to a reasonable degree.

It is difficult to compare our results to other systems which utilize a full or partial tagging dictionary, since the amount of provided knowledge is substantially different. The best comparison is to Smith and Eisner (2005) who use a partial tagging dictionary. In order to compare with their results, we projected the tagset to the coarser set of 17 that they used in their experiments. On 24K tokens, our PROTO+SIM model scored 82.2%. When Smith and Eisner (2005) limit their tagging dictionary to words which occur at least twice, their best performing neighborhood model achieves 79.5%. While these numbers seem close, for comparison, their tagging dictionary contained information about the allowable tags for 2,125 word types (out of 5,406 types) and the their system must only choose, on average, between 4.4 tags for a word. Our prototype list, however, contains information about only 116 word types and our tagger must on average choose between 16.9 tags, a much harder task. When Smith and Eisner (2005) include tagging dictionary entries for all words in the first half of their 24K tokens, giving tagging knowledge for 3,362 word types, they do achieve a higher accuracy of 88.1%.

Setting	Accuracy
BASE	46.4
PROTO	53.7
PROTO+SIM	71.5
PROTO+SIM+BOUND	74.1

Figure 7: Results on test set for ads data in (Grenager et al., 2005).

5.2 Chinese POS Tagging

We also tested our POS induction system on the Chinese POS data in the Chinese Treebank (Ircs, 2002). The model is wholly unmodified from the English version except that the suffix features are removed since, in Chinese, suffixes are not a reliable indicator of part-of-speech as in English (Tseng et al., 2005). Since we did not have access to a large auxiliary unlabeled corpus that was segmented, our distributional model was built only from the treebank text, and the distributional similarities are presumably degraded relative to the English. On 60K word tokens, BASE gave an accuracy of 34.4, PROTO gave 39.0, and PROTO+SIM gave 57.4, similar in order if not magnitude to the English case.

We believe the performance for Chinese POS tagging is not as high as English for two reasons: the general difficulty of Chinese POS tagging (Tseng et al., 2005) and the lack of a larger segmented corpus from which to build distributional models. Nonetheless, the addition of distributional similarity features does reduce the error rate by 35% from BASE.

5.3 Information Field Segmentation

We tested our framework on the CLASSIFIEDS data described in Grenager et al. (2005) under conditions similar to POS tagging. An important characteristic of this domain (see figure 1(a)) is that the hidden labels tend to be "sticky," in that fields tend to consist of runs of the same label, as in figure 1(c), in contrast with part-of-speech tagging, where we rarely see adjacent tokens given the same label. Grenager et al. (2005) report that in order to learn this "sticky" structure, they had to alter the structure of their HMM so that a fixed mass is placed on each diagonal transition. In this work, we learned this structure automatically though prototype similarity features without manually constraining the model (see

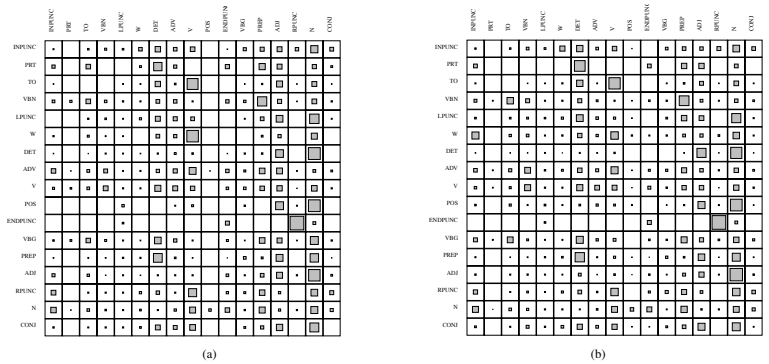


Figure 6: English coarse POS tag structure: a) corresponds to "correct" transition structure from labeled data, b) corresponds to PROTO+SIM on 24K tokens

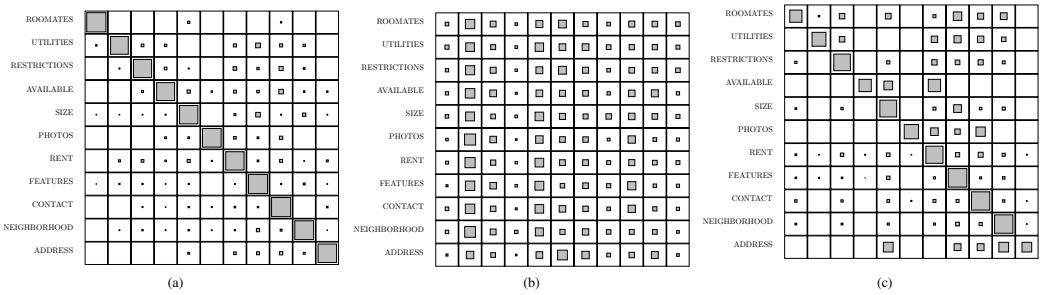


Figure 8: Field segmentation observed transition structure: (a) labeled data, (b) BASE(c) BASE+PROTO+SIM+BOUND (after post-processing)

figure 8), though we did change the similarity function (see below).

On the test set of (Grenager et al., 2005), BASE scored an accuracy of 46.4%, comparable to Grenager et al. (2005)'s unsupervised HMM baseline. Adding the prototype list (see figure 2) without distributional features yielded a slightly improved accuracy of 53.7%. For this domain, we utilized a slightly different notion of distributional similarity: we are not interested in the syntactic behavior of a word type, but its topical content. Therefore, when we collect context vectors for word types in this domain, we make no distinction by direction or distance and collect counts from a wider window. This notion of distributional similarity is more similar to latent semantic indexing (Deerwester et al., 1990). A natural consequence of this definition of distributional similarity is that many neighboring words will share the same prototypes. Therefore distributional prototype features will encourage labels to persist, naturally giving the "sticky" effect of the domain. Adding distributional similarity features to our model (PROTO+SIM) improves accuracy substantially, yielding 71.5%, a 38.4% error reduction over BASE.[6]

Another feature of this domain that Grenager et al. (2005) take advantage of is that end of sentence punctuation tends to indicate the end of a field and the beginning of a new one. Grenager et al. (2005) experiment with manually adding *boundary* states and biasing transitions from these states to not self-loop. We capture this "boundary" effect by simply adding a line to our protoype-list, adding a new BOUNDARY state (see figure 2) with a few (hand-chosen) prototypes. Since we utilize a trigram tagger, we are able to naturally capture the effect that the BOUNDARY tokens typically indicate transitions between the fields before and after the boundary token. As a post-processing step, when a token is tagged as a BOUNDARY

[6]Distributional similarity details: We collect for each word a context vector consisting of the counts for words occurring within three token occurrences of a word. We perform a SVD onto the first 50 singular vectors.

Correct Tag	Predicted Tag	% of Errors
FEATURES	SIZE	11.2
FEATURES	NBRHD	9.0
SIZE	FEATURES	7.7
NBRHD	FEATURES	6.4
ADDRESS	NBRHD	5.3
UTILITIES	FEATURES	5.3

Figure 9: Most common classified ads confusions

token it is given the same label as the previous non-BOUNDARY token, which reflects the annotational convention that boundary tokens are given the same label as the field they terminate. Adding the BOUNDARY label yields significant improvements, as indicated by the PROTO+SIM+BOUND setting in Table 5.3, surpassing the best unsupervised result of Grenager et al. (2005) which is 72.4%. Furthermore, our PROTO+SIM+BOUND model comes close to the *supervised* HMM accuracy of 74.4% reported in Grenager et al. (2005).

We also compared our method to the most basic semi-supervised setting, where fully labeled documents are provided along with unlabeled ones. Roughly 25% of the data had to be labeled in order to achieve an accuracy equal to our PROTO+SIM+BOUND model, suggesting that the use of prior knowledge in the prototype system is particularly efficient.

In table 5.3, we provide the top confusions made by our PROTO+SIM+BOUND model. As can be seen, many of our confusions involve the FEATURE field, which serves as a general purpose background state, which often differs subtly from other fields such as SIZE. For instance, the parenthical comment: *(master has walk - in closet with vanity)* is labeled as a SIZE field in the data, but our model proposed it as a FEATURE field. NEIGHBORHOOD and ADDRESS is another natural confusion resulting from the fact that the two fields share much of the same vocabulary (e.g [ADDRESS *2525 Telegraph Ave.*] vs. [NBRHD *near Telegraph*]).

Acknowledgments We would like to thank the anonymous reviewers for their comments. This work is supported by a Microsoft / CITRIS grant and by an equipment donation from Intel.

6 Conclusions

We have shown that distributional prototype features can allow one to specify a target labeling scheme in a compact and declarative way. These features give substantial error reduction on several induction tasks by allowing one to link words to prototypes according to distributional similarity. Another positive property of this approach is that it tries to reconcile the success of sequence-free distributional methods in unsupervised word clustering with the success of sequence models in supervised settings: the similarity guides the learning of the sequence model.

References

Alexander Clark. 2001. The unsupervised induction of stochastic context-free grammars using distributional clustering. In *CoNLL*.

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407.

Trond Grenager, Dan Klein, and Christopher Manning. 2005. Unsupervised learning of field segmentation models for information extraction. In *Proceedings of the 43rd Meeting of the ACL*.

Nianwen Xue Ircs. 2002. Building a large-scale annotated chinese corpus.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning (ICML)*.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Bernard Merialdo. 1991. Tagging english text with a probabilistic model. In *ICASSP*, pages 809–812.

L.R Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. In *IEEE*.

Andrew Radford. 1988. *Transformational Grammar*. Cambridge University Press, Cambridge.

Hinrich Schütze. 1995. Distributional part-of-speech tagging. In *EACL*.

Noah Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Meeting of the ACL*.

Huihsin Tseng, Daniel Jurafsky, and Christopher Manning. 2005. Morphological features help pos tagging of unknown words across language varieties. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*.

Learning Morphological Disambiguation Rules for Turkish

Deniz Yuret
Dept. of Computer Engineering
Koç University
İstanbul, Turkey
dyuret@ku.edu.tr

Ferhan Türe
Dept. of Computer Engineering
Koç University
İstanbul, Turkey
fture@ku.edu.tr

Abstract

In this paper, we present a rule based model for morphological disambiguation of Turkish. The rules are generated by a novel decision list learning algorithm using supervised training. Morphological ambiguity (e.g. lives = live+s or life+s) is a challenging problem for agglutinative languages like Turkish where close to half of the words in running text are morphologically ambiguous. Furthermore, it is possible for a word to take an unlimited number of suffixes, therefore the number of possible morphological tags is unlimited. We attempted to cope with these problems by training a separate model for each of the 126 morphological features recognized by the morphological analyzer. The resulting decision lists independently vote on each of the potential parses of a word and the final parse is selected based on our confidence on these votes. The accuracy of our model (96%) is slightly above the best previously reported results which use statistical models. For comparison, when we train a single decision list on full tags instead of using separate models on each feature we get 91% accuracy.

1 Introduction

Morphological disambiguation is the task of selecting the correct morphological parse for a given word in a given context. The possible parses of a word are generated by a morphological analyzer. In Turkish, close to half the words in running text are morphologically ambiguous. Below is a typical word "masalı" with three possible parses.

masal+Noun+A3sg+Pnon+Acc (= *the story*)
masal+Noun+A3sg+P3sg+Nom (= *his story*)
masa+Noun+A3sg+Pnon+Nomˆ DB+Adj+With (= *with tables*)

Table 1: Three parses of the word "masalı"

The first two parses start with the same root, masal (= *story, fable*), but the interpretation of the following +ı suffix is the Accusative marker in one case, and third person possessive agreement in the other. The third parse starts with a different root, masa (= *table*) followed by a derivational suffix +lı (= *with*) which turns the noun into an adjective. The symbol ˆDB represents a derivational boundary and splits the parse into chunks called inflectional groups (IGs).[1]

We will use the term *feature* to refer to individual morphological features like +Acc and +With; the term *IG* to refer to groups of features split by derivational boundaries (ˆDB), and the term *tag* to refer to the sequence of IGs following the root.

Morphological disambiguation is a useful first step for higher level analysis of any language but it is especially critical for agglutinative languages like Turkish, Czech, Hungarian, and Finnish. These languages have a relatively free constituent order, and

[1] See (Oflazer et al., 1999) for a detailed description of the morphological features used in this paper.

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 328–334,
New York, June 2006.

syntactic relations are partly determined by morphological features. Many applications including syntactic parsing, word sense disambiguation, text to speech synthesis and spelling correction depend on accurate analyses of words.

An important qualitative difference between part of speech tagging in English and morphological disambiguation in an agglutinative language like Turkish is the number of possible tags that can be assigned to a word. Typical English tag sets include less than a hundred tag types representing syntactic and morphological information. The number of potential morphological tags in Turkish is theoretically unlimited. We have observed more than ten thousand tag types in our training corpus of a million words. The high number of possible tags poses a data sparseness challenge for the typical machine learning approach, somewhat akin to what we observe in word sense disambiguation.

One way out of this dilemma could be to ignore the detailed morphological structure of the word and focus on determining only the major and minor parts of speech. However (Oflazer et al., 1999) observes that the modifier words in Turkish can have dependencies to any one of the inflectional groups of a derived word. For example, in "mavi masalı oda" (= *the room with a blue table*) the adjective "mavi" (= *blue*) modifies the noun root "masa" (= *table*) even though the final part of speech of "masalı" is an adjective. Therefore, the final part of speech and inflection of a word do not carry sufficient information for the identification of the syntactic dependencies it is involved in. One needs the full morphological analysis.

Our approach to the data sparseness problem is to consider each morphological feature separately. Even though the number of potential tags is unlimited, the number of morphological features is small: The Turkish morphological analyzer we use (Oflazer, 1994) produces tags that consist of 126 unique features. For each unique feature f, we take the subset of the training data in which one of the parses for each instance contain f. We then split this subset into positive and negative examples depending on whether the correct parse contains the feature f. These examples are used to learn rules using the Greedy Prepend Algorithm (GPA), a novel decision list learner.

To predict the tag of an unknown word, first the morphological analyzer is used to generate all its possible parses. The decision lists are then used to predict the presence or absence of each of the features contained in the candidate parses. The results are probabilistically combined taking into account the accuracy of each decision list to select the best parse. The resulting tagging accuracy is 96% on a hand tagged test set.

A more direct approach would be to train a single decision list using the full tags as the target classification. Given a word in context, such a decision list assigns a complete morphological tag instead of predicting individual morphological features. As such, it does not need the output of a morphological analyzer and should be considered a tagger rather than a disambiguator. For comparison, such a decision list was built, and its accuracy was determined to be 91% on the same test set.

The main reason we chose to work with decision lists and the GPA algorithm is their robustness to irrelevant or redundant features. The input to the decision lists include the suffixes of all possible lengths and character type information within a five word window. Each instance ends up with 40 attributes on average which are highly redundant and mostly irrelevant. GPA is able to sort out the relevant features automatically and build a fairly accurate model. Our experiments with Naive Bayes resulted in a significantly worse performance. Typical statistical approaches include the tags of the previous words as inputs in the model. GPA was able to deliver good performance without using the previous tags as inputs, because it was able to extract equivalent information implicit in the surface attributes. Finally, unlike most statistical approaches, the resulting models of GPA are human readable and open to interpretation as Section 3.1 illustrates.

The next section will review related work. Section 3 introduces decision lists and the GPA training algorithm. Section 4 presents the experiments and the results.

2 Related Work

There is a large body of work on morphological disambiguation and part of speech tagging using a variety of rule-based and statistical approaches. In the

rule-based approach a large number of hand crafted rules are used to select the correct morphological parse or POS tag of a given word in a given context (Karlsson et al., 1995; Oflazer and Tür, 1997). In the statistical approach a hand tagged corpus is used to train a probabilistic model which is then used to select the best tags in unseen text (Church, 1988; Hakkani-Tür et al., 2002). Examples of statistical and machine learning approaches that have been used for tagging include transformation based learning (Brill, 1995), memory based learning (Daelemans et al., 1996), and maximum entropy models (Ratnaparkhi, 1996). It is also possible to train statistical models using unlabeled data with the expectation maximization algorithm (Cutting et al., 1992). Van Halteren (1999) gives a comprehensive overview of syntactic word-class tagging.

Previous work on morphological disambiguation of inflectional or agglutinative languages include unsupervised learning for of Hebrew (Levinger et al., 1995), maximum entropy modeling for Czech (Hajič and Hladká, 1998), combination of statistical and rule-based disambiguation methods for Basque (Ezeiza et al., 1998), transformation based tagging for Hungarian (Megyesi, 1999).

Early work on Turkish used a constraint-based approach with hand crafted rules (Oflazer and Kuruöz, 1994). A purely statistical morphological disambiguation model was recently introduced (Hakkani-Tür et al., 2002). To counter the data sparseness problem the morphological parses are split across their derivational boundaries and certain independence assumptions are made in the prediction of each inflectional group.

A combination of three ideas makes our approach unique in the field: (1) the use of decision lists and a novel learning algorithm that combine the statistical and rule based techniques, (2) the treatment of each individual feature separately to address the data sparseness problem, and (3) the lack of dependence on previous tags and relying on surface attributes alone.

3 Decision Lists

We introduce a new method for morphological disambiguation based on decision lists. A decision list is an ordered list of rules where each rule consists of a pattern and a classification (Rivest, 1987). In our application the pattern specifies the surface attributes of the words surrounding the target such as suffixes and character types (e.g. upper vs. lower case, use of punctuation, digits). The classification indicates the presence or absence of a morphological feature for the center word.

3.1 A Sample Decision List

We will explain the rules and their patterns using the sample decision list in Table 2 trained to identify the feature `+Det` (determiner).

Rule	Class	Pattern
1	1	W=˜çok R1=+DA
2	1	L1=˜pek
3	0	W=+AzI
4	0	W=˜çok
5	1	–

Table 2: A five rule decision list for `+Det`

The value in the class column is 1 if word W should have a `+Det` feature and 0 otherwise. The pattern column describes the required attributes of the words surrounding the target word for the rule to match. The last (default) rule has no pattern, matches every instance, and assigns them `+Det`. This default rule captures the behavior of the majority of the training instances which had `+Det` in their correct parse. Rule 4 indicates a common exception: the frequently used word "çok" (meaning very) should not be assigned `+Det` by default: "çok" can be also used as an adjective, an adverb, or a postposition. Rule 1 introduces an exception to rule 4: if the right neighbor R1 ends with the suffix +DA (the locative suffix) then "çok" should receive `+Det`. The meanings of various symbols in the patterns are described below.

When the decision list is applied to a window of words, the rules are tried in the order from the most specific (rule 1) to the most general (rule 5). The first rule that matches is used to predict the classification of the center word. The last rule acts as a catch-all; if none of the other rules have matched, this rule assigns the instance a default classification. For example, the five rule decision list given above classifies the middle word in "pek çok alanda" (matches rule

W	target word	A	[ae]
L1, L2	left neighbors	I	[ıiuü]
R1, R2	right neighbors	D	[dt]
==	exact match	B	[bp]
=~	case insensitive match	C	[cç]
=+	is a suffix of	K	[kgğ]

Table 3: Symbols used in the rule patterns. Capital letters on the right represent character groups useful in identifying phonetic variations of certain suffixes, e.g. the locative suffix +DA can surface as +de, +da, +te, or +ta depending on the root word ending.

1) and "pek çok insan" (matches rule 2) as `+Det`, but "insan çok daha" (matches rule 4) as not `+Det`.

One way to interpret a decision list is as a sequence of *if-then-else* constructs familiar from programming languages. Another way is to see the last rule as the default classification, the previous rule as specifying a set of exceptions to the default, the rule before that as specifying exceptions to these exceptions and so on.

3.2 The Greedy Prepend Algorithm (GPA)

To learn a decision list from a given set of training examples the general approach is to start with a default rule or an empty decision list and keep adding the best rule to cover the unclassified or misclassified examples. The new rules can be added to the end of the list (Clark and Niblett, 1989), the front of the list (Webb and Brkic, 1993), or other positions (Newlands and Webb, 2004). Other design decisions include the criteria used to select the "best rule" and how to search for it.

The Greedy Prepend Algorithm (GPA) is a variant of the PREPEND algorithm (Webb and Brkic, 1993). It starts with a default rule that matches all instances and classifies them using the most common class in the training data. Then it keeps prepending the rule with the maximum *gain* to the front of the growing decision list until no further improvement can be made. The algorithm can be described as follows:

GPA($data$)

```
1  dlist ← NIL
2  default-class ← MOST-COMMON-CLASS(data)
3  rule ← [if TRUE then default-class]
4  while GAIN(rule, dlist, data) > 0
5      do dlist ← prepend(rule, dlist)
6         rule ← MAX-GAIN-RULE(dlist, data)
7  return dlist
```

The gain of a candidate rule in GPA is defined as the increase in the number of correctly classified instances in the training set as a result of prepending the rule to the existing decision list. This is in contrast with the original PREPEND algorithm which uses the less direct Laplace preference function (Webb and Brkic, 1993; Clark and Boswell, 1991).

To find the next rule with the maximum gain, GPA uses a heuristic search algorithm. Candidate rules are generated by adding a single new attribute to the pattern of each rule already in the decision list. The candidate with the maximum gain is prepended to the decision list and the process is repeated until no more positive gain rules can be found. Note that if the best possible rule has more than one extra attribute compared to the existing rules in the decision list, a suboptimal rule will be selected. The original PREPEND uses an admissible search algorithm, OPUS, which is guaranteed to find the best possible candidate (Webb, 1995), but we found OPUS to be too slow to be practical for a problem of this scale.

We picked GPA for the morphological disambiguation problem because we find it to be fast and fairly robust to the existence of irrelevant or redundant attributes. The average training instance has 40 attributes describing the suffixes of all possible lengths and character type information in a five word window. Most of this information is redundant or irrelevant to the problem at hand. The number of distinct attributes is on the order of the number of distinct word-forms in the training set. Nevertheless GPA is able to process a million training instances for each of the 126 unique morphological features and produce a model with state of the art accuracy in about two hours on a regular desktop PC.[2]

[2]Pentium 4 CPU 2.40GHz

4 Experiments and Results

In this section we present the details of the data, the training and testing procedures, the surface attributes used, and the accuracy results.

4.1 Training Data

documents	2383
sentences	50673
tokens	948404
parses	1.76 per token
IGs	1.33 per parse
features	3.29 per IG
unique tokens	111467
unique tags	11084
unique IGs	2440
unique features	126
ambiguous tokens	399223 (42.1%)

Table 4: Statistics for the training data

Our training data consists of about 1 million words of semi-automatically disambiguated Turkish news text. For each one of the 126 unique morphological features, we used the subset of the training data in which instances have the given feature in at least one of their generated parses. We then split this subset into positive and negative examples depending on whether the correct parse contains the given feature. A decision list specific to that feature is created using GPA based on these examples.

Some relevant statistics for the training data are given in Table 4.

4.2 Input Attributes

Once the training data is selected for a particular morphological feature, each instance is represented by surface attributes of five words centered around the target word. We have tried larger window sizes but no significant improvement was observed. The attributes computed for each word in the window consist of the following:

1. The exact word string (e.g. W==Ali'nin)
2. The lowercase version (e.g. W=˜ali'nin) Note: all digits are replaced by 0's at this stage.
3. All suffixes of the lowercase version (e.g. W=+n, W=+In, W=+nIn, W=+'nIn, etc.) Note: certain characters are replaced with capital letters representing character groups mentioned in Table 3. These groups help the algorithm recognize different forms of a suffix created by the phonetic rules of Turkish: for example the locative suffix +DA can surface as +de, +da, +te, or +ta depending on the ending of the root word.
4. Attributes indicating the types of characters at various positions of the word (e.g. Ali'nin would be described with W=UPPER-FIRST, W=LOWER-MID, W=APOS-MID, W=LOWER-LAST)

Each training instance is represented by 40 attributes on average. The GPA procedure is responsible for picking the attributes that are relevant to the decision. No dictionary information is required or used, therefore the models are fairly robust to unknown words. One potentially useful source of attributes is the tags assigned to previous words which we plan to experiment with in future work.

4.3 The Decision Lists

At the conclusion of the training, 126 decision lists are produced of the form given in Table 2. The number of rules in each decision list range from 1 to 6145. The longer decision lists are typically for part of speech features, e.g. distinguishing nouns from adjectives, and contain rules specific to lexical items. The average number of rules is 266. To get an estimate on the accuracy of each decision list, we split the one million word data into training, validation, and test portions using the ratio 4:1:1. The training set accuracy of the decision lists is consistently above 98%. The test set accuracies of the 126 decision lists range from 80% to 100% with the average at 95%. Table 5 gives the six worst features with test set accuracy below 89%; these are the most difficult to disambiguate.

4.4 Correct Tag Selection

To evaluate the candidate tags, we need to combine the results of the decision lists. We assume that the presence or absence of each feature is an independent event with a probability determined by the test set accuracy of the corresponding decision list. For example, if the `+P3pl` decision list predicts YES, we assume that the `+P3pl` feature is present with

87.89%	+Acquire	To acquire (noun)
86.18%	+PCIns	Postposition subcat.
85.11%	+Fut	Future tense
84.08%	+P3pl	3. plural possessive
80.79%	+Neces	Must
79.81%	+Become	To become (noun)

Table 5: The six features with the worst test set accuracy.

probability 0.8408 (See Table 5). If the +Fut decision list predicts NO, we assume the +Fut feature is present with probability $1 - 0.8511 = 0.1489$. To avoid zero probabilities we cap the test set accuracies at 99%.

Each candidate tag indicates the presence of certain features and the absence of others. The probability of the tag being correct under our independence assumption is the product of the probabilities for the presence and absence of each of the 126 features as determined by our decision lists. For efficiency, one can neglect the features that are absent from all the candidate tags because their contribution will not effect the comparison.

4.5 Results

The final evaluation of the model was performed on a test data set of 958 instances. The possible parses for each instance were generated by the morphological analyzer and the correct one was picked manually. 40% of the instances were ambiguous, which on the average had 3.9 parses. The disambiguation accuracy of our model was 95.82%. The 95% confidence interval for the accuracy is [0.9457, 0.9708].

An analysis of the mistakes in the test data show that at least some of them are due to incorrect tags in our training data. The training data was semi-automatically generated and thus contained some errors. Based on hand evaluation of the differences between the training data tags and the GPA generated tags, we estimate the accuracy of the training data to be below 95%. We ran two further experiments to see if we could improve on the initial results.

In our first experiment we used our original model to re-tag the training data. The re-tagged training data was used to construct a new model. The resulting accuracy on the test set increased to 96.03%, not a statistically significant improvement.

In our second experiment we used only unambiguous instances for training. Decision list training requires negative examples, so we selected random unambiguous instances for positive and negative examples for each feature. The accuracy of the resulting model on the test set was 82.57%. The problem with selecting unambiguous instances is that certain common disambiguation decisions are never represented during training. More careful selection of negative examples and a sophisticated bootstrapping mechanism may still make this approach workable.

Finally, we decided to see if our decision lists could be used for tagging rather than disambiguation, i.e. given a word in a context decide on the full tag *without the help of a morphological analyzer*. Even though the number of possible tags is unlimited, the most frequent 1000 tags cover about 99% of the instances. A single decision list trained with the full tags was able to achieve 91.23% accuracy using 10000 rules. This is a promising result and will be explored further in future work.

5 Contributions

We have presented an automated approach to learn morphological disambiguation rules for Turkish using a novel decision list induction algorithm, GPA. The only input to the rules are the surface attributes of a five word window. The approach can be generalized to other agglutinative languages which share the common challenge of a large number of potential tags. Our approach for resolving the data sparseness problem caused by the large number of tags is to generate a separate model for each morphological feature. The predictions for individual features are probabilistically combined based on the accuracy of each model to select the best tag. We were able to achieve an accuracy around 96% using this approach.

Acknowledgments

We would like to thank Kemal Oflazer of Sabancı University for providing us with the Turkish morphological analyzer, training and testing data for disambiguation, and valuable feedback.

References

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

Church, K. W. (1988). A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing*, pages 136–143.

Clark, P. and Boswell, R. (1991). Rule induction with CN2: Some recent improvements. In Kodratoff, Y., editor, *Machine Learning – Proceedings of the Fifth European Conference (EWSL-91)*, pages 151–163, Berlin. Springer-Verlag.

Clark, P. and Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, 3:261–283.

Cutting, D., Kupiec, J., Pedersen, J., and Sibun, P. (1992). A practical part-of-speech tagger. In *Proceedings of the 3rd Conference on Applied Language Processing*, pages 133–140.

Daelemans, W. et al. (1996). MBT: A memory-based part of speech tagger-generator. In Ejerhead, E. and Dagan, I., editors, *Proceedings of the Fourth Workshop on Very Large Corpora*, pages 14–27.

Ezeiza, N. et al. (1998). Combining stochastic and rule-based methods for disambiguation in agglutinative languages. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics (COLING/ACL98)*, pages 379–384.

Hajič, J. and Hladká, B. (1998). Tagging inflective languages: Prediction of morphological categories for a rich, structured tagset. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics (COLING/ACL98)*, pages 483–490, Montreal, Canada.

Hakkani-Tür, D. Z., Oflazer, K., and Tür, G. (2002). Statistical morphological disambiguation for agglutinative languages. *Computers and the Humanities*, 36:381–410.

Karlsson, F., Voutialinen, A., Heikkilä, J., and Anttila, A. (1995). *Constraint Grammar - A Language Independent System for Parsing Unrestricted Text*. Mouton de Gruyter.

Levinger, M., Ornan, U., and Itai, A. (1995). Learning morpho-lexical probabilities from an untagged corpus with an application to hebrew. *Computational Linguistics*, 21(3):383–404.

Megyesi, B. (1999). Improving brill's pos tagger for an agglutinative language. In Pascale, F. and Joe, Z., editors, *Proceedings of the Joing SIGDAT Conference on Empirical Methods in Natural Language and Very Large Corpora*, pages 275–284, College Park, Maryland, USA.

Newlands, D. and Webb, G. I. (2004). Alternative strategies for decision list construction. In *Proceedings of the Fourth Data Mining Conference (DM IV 03)*, pages 265–273.

Oflazer, K. (1994). Two-level description of turkish morphology. *Literary and Linguistic Computing*, 9(2):137–148.

Oflazer, K., Hakkani-Tür, D. Z., and Tür, G. (1999). Design for a turkish treebank. In *Proceedings of the Workshop on Linguistically Interpreted Corpora, EACL 99*, Bergen, Norway.

Oflazer, K. and Kuruöz, İ. (1994). Tagging and morphological disambiguation of turkish text. In *Proceedings of the 4th Applied Natural Language Processing Conference*, pages 144–149. ACL.

Oflazer, K. and Tür, G. (1997). Morphological disambiguation by voting constraints. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL97, EACL97)*, Madrid, Spain.

Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Rivest, R. L. (1987). Learning decision lists. *Machine Learning*, 2:229–246.

van Halteren, H., editor (1999). *Syntactic Wordclass Tagging*. Text, Speech and Language Technology. Kluwer Academic Publishers.

Webb, G. I. (1995). Opus: An efficient admissible algorithm for unordered search. *JAIR*, 3:431–465.

Webb, G. I. and Brkic, N. (1993). Learning decision lists by prepending inferred rules. In *Proceedings of the AI 93 Workshop on Machine Learning and Hybrid Systems*, pages 6–10, Melbourne.

Cross-Entropy and Estimation of Probabilistic Context-Free Grammars

Anna Corazza
Department of Physics
University "Federico II"
via Cinthia
I-80126 Napoli, Italy
corazza@na.infn.it

Giorgio Satta
Department of Information Engineering
University of Padua
via Gradenigo, 6/A
I-35131 Padova, Italy
satta@dei.unipd.it

Abstract

We investigate the problem of training probabilistic context-free grammars on the basis of a distribution defined over an infinite set of trees, by minimizing the cross-entropy. This problem can be seen as a generalization of the well-known maximum likelihood estimator on (finite) tree banks. We prove an unexpected theoretical property of grammars that are trained in this way, namely, we show that the derivational entropy of the grammar takes the same value as the cross-entropy between the input distribution and the grammar itself. We show that the result also holds for the widely applied maximum likelihood estimator on tree banks.

1 Introduction

Probabilistic context-free grammars are able to describe hierarchical, tree-shaped structures underlying sentences, and are widely used in statistical natural language processing; see for instance (Collins, 2003) and references therein. Probabilistic context-free grammars seem also more suitable than finite-state devices for language modeling, and several language models based on these grammars have been recently proposed in the literature; see for instance (Chelba and Jelinek, 1998), (Charniak, 2001) and (Roark, 2001).

Empirical estimation of probabilistic context-free grammars is usually carried out on tree banks, that is, finite samples of parse trees, through the maximization of the likelihood of the sample itself. It is well-known that this method also minimizes the cross-entropy between the probability distribution induced by the tree bank, also called the empirical distribution, and the tree probability distribution induced by the estimated grammar.

In this paper we generalize the maximum likelihood method, proposing an estimation technique that works on any unrestricted tree distribution defined over an infinite set of trees. This generalization is theoretically appealing, and allows us to prove unexpected properties of the already mentioned maximum likelihood estimator for tree banks, that were not previously known in the literature on statistical natural language parsing. More specifically, we investigate the following information theoretic quantities

- the cross-entropy between the unrestricted tree distribution given as input and the tree distribution induced by the estimated probabilistic context-free grammar; and
- the derivational entropy of the estimated probabilistic context-free grammar.

These two quantities are usually unrelated. We show that these two quantities take the same value when the probabilistic context-free grammar is trained using the minimal cross-entropy criterion. We then translate back this property to the method of maximum likelihood estimation. Our general estimation method also has practical applications in cases one uses a probabilistic context-free grammar to approximate strictly more powerful rewriting systems,

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 335–342,
New York, June 2006.

as for instance probabilistic tree adjoining grammars (Schabes, 1992).

Not much is found in the literature about the estimation of probabilistic grammars from infinite distributions. This line of research was started in (Nederhof, 2005), investigating the problem of training an input probabilistic finite automaton from an infinite tree distribution specified by means of an input probabilistic context-free grammar. The problem we consider in this paper can then be seen as a generalization of the above problem, where the input model to be trained is a probabilistic context-free grammar and the input distribution is an unrestricted tree distribution. In (Chi, 1999) an estimator that maximizes the likelihood of a probability distribution defined over a finite set of trees is introduced, as a generalization of the maximum likelihood estimator. Again, the problems we consider here can be thought of as generalizations of such estimator to the case of distributions over infinite sets of trees or sentences.

The remainder of this paper is structured as follows. Section 2 introduces the basic notation and definitions and Section 3 discusses our new estimation method. Section 4 presents our main result, which is transferred in Section 5 to the method of maximum likelihood estimation. Section 6 discusses some simple examples, and Section 7 closes with some further discussion.

2 Preliminaries

Throughout this paper we use standard notation and definitions from the literature on formal languages and probabilistic grammars, which we briefly summarize below. We refer the reader to (Hopcroft and Ullman, 1979) and (Booth and Thompson, 1973) for a more precise presentation.

A **context-free grammar** (CFG) is a tuple $G = (N, \Sigma, R, S)$, where N is a finite set of nonterminal symbols, Σ is a finite set of terminal symbols disjoint from N, $S \in N$ is the start symbol and R is a finite set of rules. Each rule has the form $A \to \alpha$, where $A \in N$ and $\alpha \in (\Sigma \cup N)^*$. We denote by $L(G)$ and $T(G)$ the set of all strings, resp., trees, generated by G. For $t \in T(G)$, the yield of t is denoted by $y(t)$.

For a nonterminal A and a string α, we write $f(A, \alpha)$ to denote the number of occurrences of A in α. For a rule $(A \to \alpha) \in R$ and a tree $t \in T(G)$, $f(A \to \alpha, t)$ denotes the number of occurrences of $A \to \alpha$ in t. We let $f(A, t) = \sum_\alpha f(A \to \alpha, t)$.

A **probabilistic** context-free grammar (PCFG) is a pair $\mathcal{G} = (G, p_G)$, with G a CFG and p_G a function from R to the real numbers in the interval $[0, 1]$. A PCFG is **proper** if for every $A \in N$ we have $\sum_\alpha p_G(A \to \alpha) = 1$. The probability of $t \in T(G)$ is the product of the probabilities of all rules in t, counted with their multiplicity, that is,

$$p_G(t) \;=\; \prod_{A \to \alpha} p_G(A \to \alpha)^{f(A \to \alpha, t)}. \quad (1)$$

The probability of $w \in L(G)$ is the sum of the probabilities of all the trees that generate w, that is,

$$p_G(w) \;=\; \sum_{y(t) = w} p_G(t). \quad (2)$$

A PCFG is **consistent** if $\sum_{t \in T(G)} p_G(t) = 1$.

In this paper we write $\log$ for logarithms in base 2 and $\ln$ for logarithms in the natural base e. We also assume $0 \cdot \log 0 = 0$. We write E_p to denote the expectation operator under distribution p. In case $\mathcal{G}$ is proper and consistent, we can define the **derivational entropy** of $\mathcal{G}$ as the expectation of the information of parse trees in $T(G)$, computed under distribution p_G, that is,

$$\begin{aligned} H_d(p_G) \;&=\; E_{p_G} \log \frac{1}{p_G(t)} \\ &=\; -\sum_{t \in T(G)} p_G(t) \cdot \log p_G(t). \end{aligned} \quad (3)$$

Similarly, for each $A \in N$ we also define the **nonterminal entropy** of A as

$$\begin{aligned} &H_A(p_G) = \\ &\quad= E_{p_G} \log \frac{1}{p_G(A \to \alpha)} \\ &\quad= -\sum_{\alpha} p_G(A \to \alpha) \cdot \log p_G(A \to \alpha). \end{aligned} \quad (4)$$

3 Estimation based on cross-entropy

Let T be an infinite set of (finite) trees with internal nodes labeled by symbols in N, root nodes labeled by $S \in N$ and leaf nodes labeled by symbols

in Σ. We assume that the set of rules that are observed in the trees in T is drawn from some finite set R. Let p_T be a probability distribution defined over T, that is, a function from T to set $[0,1]$ such that $\sum_{t\in T} p_T(t) = 1$.

The **skeleton** CFG underlying T is defined as $G = (N, \Sigma, R, S)$. Note that we have $T \subseteq T(G)$ and, in the general case, there might be trees in $T(G)$ that do not appear in T. We wish anyway to approximate distribution p_T the best we can, by turning G into some proper PCFG $\mathcal{G} = (G, p_G)$ and setting parameters $p_G(A \to \alpha)$ appropriately, for each $(A \to \alpha) \in R$.

One possible criterion is to choose p_G in such a way that the cross-entropy between p_T and p_G is minimized, where we now view p_G as a probability distribution defined over $T(G)$. The **cross-entropy** between p_T and p_G is defined as the expectation under distribution p_T of the information, computed under distribution p_G, of the trees in $T(G)$

$$\begin{aligned} H(p_T \,||\, p_G) &= E_{p_T} \log \frac{1}{p_G(t)} \\ &= -\sum_{t\in T} p_T(t) \cdot \log p_G(t). \quad (5) \end{aligned}$$

Since $\mathcal{G}$ should be proper, the minimization of (5) is subject to the constraints $\sum_\alpha p_G(A \to \alpha) = 1$, for each $A \in N$.

To solve the minimization problem above, we use Lagrange multipliers λ_A for each $A \in N$ and define the form

$$\begin{aligned} \nabla &= \sum_{A\in N} \lambda_A \cdot (\sum_\alpha p_G(A \to \alpha) - 1) + \\ &\quad -\sum_{t\in T} p_T(t) \cdot \log p_G(t). \quad (6) \end{aligned}$$

We now view ∇ as a function of all the λ_A and the $p_G(A \to \alpha)$, and consider all the partial derivatives of ∇. For each $A \in N$ we have

$$\frac{\partial \nabla}{\partial \lambda_A} = \sum_\alpha p_G(A \to \alpha) - 1.$$

For each $(A \to \alpha) \in R$ we have

$$\begin{aligned} &\frac{\partial \nabla}{\partial p_G(A \to \alpha)} = \\ &= \lambda_A - \frac{\partial}{\partial p_G(A \to \alpha)} \sum_{t\in T} p_T(t) \cdot \log p_G(t) \\ &= \lambda_A - \sum_{t\in T} p_T(t) \cdot \frac{\partial}{\partial p_G(A \to \alpha)} \log p_G(t) \\ &= \lambda_A - \sum_{t\in T} p_T(t) \cdot \frac{\partial}{\partial p_G(A \to \alpha)} \\ &\quad \log \prod_{(B\to\beta)\in R} p_G(B \to \beta)^{f(B\to\beta,t)} \\ &= \lambda_A - \sum_{t\in T} p_T(t) \cdot \frac{\partial}{\partial p_G(A \to \alpha)} \\ &\quad \sum_{(B\to\beta)\in R} f(B \to \beta, t) \cdot \log p_G(B \to \beta) \\ &= \lambda_A - \sum_{t\in T} p_T(t) \cdot \sum_{(B\to\beta)\in R} f(B \to \beta, t) \cdot \\ &\quad \frac{\partial}{\partial p_G(A \to \alpha)} \log p_G(B \to \beta) \\ &= \lambda_A - \sum_{t\in T} p_T(t) \cdot f(A \to \alpha, t) \cdot \\ &\quad \cdot \frac{1}{\ln(2)} \cdot \frac{1}{p_G(A \to \alpha)} \\ &= \lambda_A - \frac{1}{\ln(2)} \cdot \frac{1}{p_G(A \to \alpha)} \cdot \\ &\quad \cdot \sum_{t\in T} p_T(t) \cdot f(A \to \alpha, t) \\ &= \lambda_A - \frac{1}{\ln(2)} \cdot \frac{1}{p_G(A \to \alpha)} \cdot \\ &\quad \cdot E_{p_T} f(A \to \alpha, t). \end{aligned}$$

We now need to solve a system of $|N|+|R|$ equations obtained by setting to zero all of the above partial derivatives. From each equation $\frac{\partial \nabla}{\partial p_G(A\to\alpha)} = 0$ we obtain

$$\begin{aligned} &\lambda_A \cdot \ln(2) \cdot p_G(A \to \alpha) = \\ &\quad = E_{p_T} f(A \to \alpha, t). \quad (7) \end{aligned}$$

We sum over all strings α such that $(A \to \alpha) \in R$

$$\begin{aligned} &\lambda_A \cdot \ln(2) \cdot \sum_\alpha p_G(A \to \alpha) = \\ &\quad = \sum_\alpha E_{p_T} f(A \to \alpha, t) \\ &\quad = \sum_\alpha \sum_{t\in T} p_T(t) \cdot f(A \to \alpha, t) \\ &\quad = \sum_{t\in T} p_T(t) \cdot \sum_\alpha f(A \to \alpha, t) \\ &\quad = \sum_{t\in T} p_T(t) \cdot f(A, t) \\ &\quad = E_{p_T} f(A, t). \quad (8) \end{aligned}$$

From each equation $\frac{\partial \nabla}{\partial \lambda_A} = 0$ we obtain $\sum_\alpha p_G(A \to \alpha) = 1$ for each $A \in N$ (our original constraints). Combining with (8) we obtain

$$\lambda_A \cdot \ln(2) \;=\; E_{p_T}\, f(A,t). \tag{9}$$

Replacing (9) into (7) we obtain, for every rule $(A \to \alpha) \in R$,

$$p_G(A \to \alpha) \;=\; \frac{E_{p_T}\, f(A \to \alpha, t)}{E_{p_T}\, f(A,t)}. \tag{10}$$

The equations in (10) define the desired estimator for our PCFG, assigning to each rule $A \to \alpha$ a probability specified as the ratio between the expected number of $A \to \alpha$ and the expected number of A, under the distribution p_T. We remark here that the minimization of the cross-entropy above is equivalent to the minimization of the Kullback-Leibler distance between p_T and p_G, viewed as tree distributions. Also, note that the likelihood of an infinite set of derivations would always be zero and therefore cannot be considered here.

To be used in the next section, we now show that the PCFG $\mathcal{G}$ obtained as above is consistent. The line of our argument below follows a proof provided in (Chi and Geman, 1998) for the maximum likelihood estimator based on finite tree distributions. Without loss of generality, we assume that in $\mathcal{G}$ the start symbol S is never used in the right-hand side of a rule.

For each $A \in N$, let q_A be the probability that a derivation in $\mathcal{G}$ rooted in A fails to terminate. We can then write

$$q_A \;\leq\; \sum_{B \in N} q_B \cdot \sum_\alpha\, p_G(A \to \alpha) f(B,\alpha). \tag{11}$$

The inequality follows from the fact that the events considered in the right-hand side of (11) are not mutually exclusive. Combining (10) and (11) we obtain

$$\begin{aligned} & q_A \cdot E_{p_T}\, f(A,t) \leq \\ & \quad \leq \sum_{B \in N} q_B \cdot \sum_\alpha\, E_{p_T} f(A \to \alpha, t) f(B,\alpha). \end{aligned}$$

Summing over all nonterminals we have

$$\begin{aligned} & \sum_{A \in N} q_A \cdot E_{p_T}\, f(A,t) \leq \\ & \quad \leq \sum_{B \in N} q_B \cdot \sum_{A \in N} \sum_\alpha\, E_{p_T}\, f(A \to \alpha, t) f(B,\alpha) \\ & \quad = \sum_{B \in N} q_B \cdot E_{p_T}\, f_c(B,t), \end{aligned} \tag{12}$$

where $f_c(B,t)$ indicates the number of times a node labeled by nonterminal B appears in the derivation tree t as a child of some other node.

From our assumptions on the start symbol S, we have that S only appears at the root of the trees in $T(G)$. Then it is easy to see that, for every $A \neq S$, we have $E_{p_T} f_c(A,t) = E_{p_T} f(A,t)$, while $E_{p_T} f_c(S,t) = 0$ and $E_{p_T} f(S,t) = 1$. Using these relations in (12) we obtain

$$q_S \cdot E_{p_T}\, f(S,T) \;\leq\; q_S \cdot E_{p_T}\, f_c(S,T),$$

from which we conclude $q_S = 0$, thus implying the consistency of $\mathcal{G}$.

4 Cross-entropy and derivational entropy

In this section we present the main result of the paper. We show that, when $\mathcal{G} = (G, p_G)$ is estimated by minimizing the cross-entropy in (5), then such cross-entropy takes the same value as the derivational entropy of $\mathcal{G}$, defined in (3).

In (Nederhof and Satta, 2004) relations are derived for the exact computation of $H_d(p_G)$. For later use, we report these relations below, under the assumption that $\mathcal{G}$ is consistent (see Section 3). We have

$$H_d(p_G) \;=\; \sum_{A \in N} out_{\mathcal{G}}(A) \cdot H_A(p_G). \tag{13}$$

Quantities $H_A(p_G)$, $A \in N$, have been defined in (4). For each $A \in N$, quantity $out_{\mathcal{G}}(A)$ is the sum of the probabilities of all trees generated by $\mathcal{G}$, having root labeled by S and having a yield composed of terminal symbols with an unexpanded occurrence of nonterminal A. Again, we assume that symbol S does not appear in any of the right-hand sides of the rules in R. This means that S only appears at the root of the trees in $T(G)$. Under this condition, quantities $out_{\mathcal{G}}(A)$ can be exactly computed by solving the following system of linear equations (see also (Nederhof, 2005))

$$out_{\mathcal{G}}(S) \;=\; 1; \tag{14}$$

for each $A \neq S$

$$\begin{aligned} & out_{\mathcal{G}}(A) = \\ & \quad = \sum_{B \to \beta} out_{\mathcal{G}}(B) \cdot f(A,\beta) \cdot p_G(B \to \beta) \end{aligned} \tag{15}$$

We can now prove the equality

$$H_d(p_G) \;=\; H(p_T \,||\, p_G), \tag{16}$$

where $\mathcal{G}$ is the PCFG estimated by minimizing the cross-entropy in (5), as described in Section 3.

We start from the definition of cross-entropy

$$\begin{aligned}
&H(p_T \,||\, p_G) = \\
&= -\sum_{t\in T} p_T(t)\cdot \log p_G(t) \\
&= -\sum_{t\in T} p_T(t)\cdot \log \prod_{A\to\alpha} p_G(A\to\alpha)^{f(A\to\alpha,t)} \\
&= -\sum_{t\in T} p_T(t)\cdot \\
&\qquad \cdot \sum_{A\to\alpha} f(A\to\alpha,t)\cdot \log p_G(A\to\alpha) \\
&= -\sum_{A\to\alpha} \log p_G(A\to\alpha)\cdot \\
&\qquad \cdot \sum_{t\in T} p_T(t)\cdot f(A\to\alpha,t) \\
&= -\sum_{A\to\alpha} \log p_G(A\to\alpha)\cdot \\
&\qquad \cdot E_{p_T}\, f(A\to\alpha,t).
\end{aligned} \tag{17}$$

From our estimator in (10) we can write

$$\begin{aligned}
&E_{p_T}\, f(A\to\alpha,t) = \\
&\quad = p_G(A\to\alpha)\cdot E_{p_T}\, f(A,t).
\end{aligned} \tag{18}$$

Replacing (18) into (17) gives

$$\begin{aligned}
&H(p_T \,||\, p_G) = \\
&= -\sum_{A\to\alpha} \log p_G(A\to\alpha)\cdot \\
&\qquad \cdot p_G(A\to\alpha)\cdot E_{p_T}\, f(A,t) \\
&= -\sum_{A\in N} E_{p_T}\, f(A,t)\cdot \\
&\qquad \cdot \sum_{\alpha} p_G(A\to\alpha)\cdot \log p_G(A\to\alpha) \\
&= \sum_{A\in N} E_{p_T}\, f(A,t)\cdot H(p_G,A).
\end{aligned} \tag{19}$$

Comparing (19) with (13) we see that, in order to prove the equality in (16), we need to show relations

$$E_{p_T}\, f(A,t) \;=\; out_{\mathcal{G}}(A), \tag{20}$$

for every $A \in N$. We have already observed in Section 3 that, under our assumption on the start symbol S, we have

$$E_{p_T}\, f(S,t) \;=\; 1. \tag{21}$$

We now observe that, for any $A \in N$ with $A \neq S$ and any $t \in T(G)$, we have

$$\begin{aligned}
&f(A,t) = \\
&\quad = \sum_{B\to\beta} f(B\to\beta,t)\cdot f(A,\beta).
\end{aligned} \tag{22}$$

For each $A \in N$ with $A \neq S$ we can then write

$$\begin{aligned}
&E_{p_T}\, f(A,t) = \\
&\quad = \sum_{t\in T} p_T(t)\cdot f(A,t) \\
&\quad = \sum_{t\in T} p_T(t)\cdot \sum_{B\to\beta} f(B\to\beta,t)\cdot f(A,\beta) \\
&\quad = \sum_{B\to\beta}\sum_{t\in T} p_T(t)\cdot f(B\to\beta,t)\cdot f(A,\beta) \\
&\quad = \sum_{B\to\beta} E_{p_T}\, f(B\to\beta,t)\cdot f(A,\beta).
\end{aligned} \tag{23}$$

Once more we use relation (18), which replaced in (23) provides

$$\begin{aligned}
&E_{p_T}\, f(A,t) = \\
&\quad = \sum_{B\to\beta} E_{p_T}\, f(B,t)\cdot \\
&\qquad \cdot f(A,\beta)\cdot p_G(B\to\beta).
\end{aligned} \tag{24}$$

Notice that the linear system in (14) and (15) and the linear system in (21) and (24) are the same. Thus we conclude that quantities $E_{p_T}\, f(A,t)$ and $out_{\mathcal{G}}(A)$ are the same for each $A \in N$. This completes our proof of the equality in (16). Some examples will be discussed in Section 6.

Besides its theoretical significance, the equality in (16) can also be exploited in the computation of the cross-entropy in practical applications. In fact, cross-entropy is used as a measure of tightness in comparing different models. In case of estimation from an infinite distribution p_T, the definition of the cross-entropy $H(p_T \,||\, p_G)$ contains an infinite summation, which is problematic for the computation of such quantity. In standard practice, this problem is overcome by generating a finite sample $T^{(n)}$ of large size n, through the distribution p_T, and then computing the approximation (Manning and Schütze, 1999)

$$H(p_T \,||\, p_G) \;\sim\; -\frac{1}{n}\sum_{t\in T} f(t,T^{(n)})\cdot \log p_G(t),$$

where $f(t, T^{(n)})$ indicates the multiplicity, that is, the number of occurrences, of t in $T^{(n)}$. However, in practical applications n must be very large in order to have a small error. Based on the results in this section, we can instead compute the exact value of $H(p_T \,||\, p_G)$ by computing the derivational entropy $H_d(p_G)$, using relation (13) and solving the linear system in (14) and (15), which takes cubic time in the number of nonterminals of the grammar.

5 Estimation based on likelihood

In natural language processing applications, the estimation of a PCFG is usually carried out on the basis of a finite sample of trees, called tree bank. The so-called maximum likelihood estimation (MLE) method is exploited, which maximizes the likelihood of the observed data. In this section we show that the MLE method is a special case of the estimation method presented in Section 3, and that the results of Section 4 also hold for the MLE method.

Let $\mathcal{T}$ be a tree sample, and let T be the underlying set of trees. For $t \in T$, we let $f(t, \mathcal{T})$ be the multiplicity of t in $\mathcal{T}$. We define

$$\begin{aligned} & f(A \rightarrow \alpha, \mathcal{T}) = \\ & \quad = \sum_{t \in T} f(t, \mathcal{T}) \cdot f(A \rightarrow \alpha, t), \end{aligned} \tag{25}$$

and let $f(A, \mathcal{T}) = \sum_\alpha f(A \rightarrow \alpha, \mathcal{T})$. We can induce from $\mathcal{T}$ a probability distribution $p_\mathcal{T}$, defined over T, by letting for each $t \in T$

$$p_\mathcal{T}(t) \quad = \quad \frac{f(t, \mathcal{T})}{|\mathcal{T}|}. \tag{26}$$

Note that $\sum_{t \in T} p_\mathcal{T}(t) = 1$. Distribution $p_\mathcal{T}$ is called the **empirical distribution** of $\mathcal{T}$.

Assume that the trees in T have internal nodes labeled by symbols in N, root nodes labeled by S and leaf nodes labeled by symbols in Σ. Let also R be the finite set of rules that are observed in $\mathcal{T}$. We define the skeleton CFG underlying T as $G = (N, \Sigma, R, S)$. In the MLE method we probabilistically extend the skeleton CFG G by means of a function p_G that maximizes the likelihood of $\mathcal{T}$, defined as

$$p_G(\mathcal{T}) \quad = \quad \prod_{t \in T} p_G(t)^{f(t, \mathcal{T})}, \tag{27}$$

subject to the usual properness conditions on p_G. Such maximization provides the estimator (see for instance (Chi and Geman, 1998))

$$p_G(A \rightarrow \alpha) \quad = \quad \frac{f(A \rightarrow \alpha, \mathcal{T})}{f(A, \mathcal{T})}. \tag{28}$$

Let us consider the estimator in (10). If we replace distribution p_T with our empirical distribution $p_\mathcal{T}$, we derive

$$\begin{aligned} & p_G(A \rightarrow \alpha) = \\ & \quad = \frac{E_{p_\mathcal{T}}\ f(A \rightarrow \alpha, t)}{E_{p_\mathcal{T}}\ f(A, t)} \\ & \quad = \frac{\sum_{t \in T}\ \frac{f(t, \mathcal{T})}{|\mathcal{T}|} \cdot f(A \rightarrow \alpha, t)}{\sum_{t \in T}\ \frac{f(t, \mathcal{T})}{|\mathcal{T}|} \cdot f(A, t)} \\ & \quad = \frac{\sum_{t \in T}\ f(t, \mathcal{T}) \cdot f(A \rightarrow \alpha, t)}{\sum_{t \in T}\ f(t, \mathcal{T}) \cdot f(A, t)} \\ & \quad = \frac{f(A \rightarrow \alpha, \mathcal{T})}{f(A, \mathcal{T})}. \end{aligned} \tag{29}$$

This is precisely the estimator in (28).

From relation (29) we conclude that the MLE method can be seen as a special case of the general estimator in Section 3, with the input distribution defined over a finite set of trees. We can also derive the well-known fact that, in the finite case, the maximization of the likelihood $p_G(\mathcal{T})$ corresponds to the minimization of the cross-entropy $H(p_\mathcal{T} \,||\, p_G)$.

Let now $\mathcal{G} = (G, p_G)$ be a PCFG trained on $\mathcal{T}$ using the MLE method. Again from relation (29) and Section 3 we have that $\mathcal{G}$ is consistent. This result has been firstly shown in (Chaudhuri et al., 1983) and later, with a different proof technique, in (Chi and Geman, 1998). We can then transfer the results of Section 4 to the supervised MLE method, showing the equality

$$H_d(p_G) \quad = \quad H(p_\mathcal{T} \,||\, p_G). \tag{30}$$

This result was not previously known in the literature on statistical parsing of natural language. Some examples will be discussed in Section 6.

6 Some examples

In this section we discuss a simple example with the aim of clarifying the theoretical results in the previous sections. For a real number q with $0 < q < 1$,

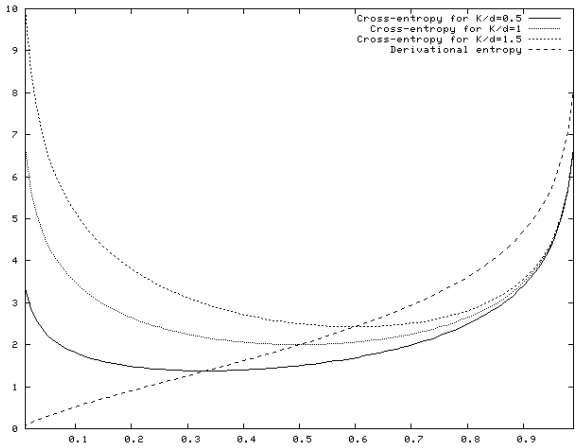


Figure 1: Derivational entropy of $\mathcal{G}_q$ and cross-entropies for three different corpora.

consider the CFG G defined by the two rules $S \rightarrow aS$ and $S \rightarrow a$, and let $\mathcal{G}_q = (G, p_{G,q})$ be the probabilistic extension of G with $p_{G,q}(S \rightarrow aS) = q$ and $p_{G,q}(S \rightarrow a) = 1 - q$. This grammar is unambiguous and consistent, and each tree t generated by G has probability $p_{G,q}(t) = q^i \cdot (1-q)$, where $i \geq 0$ is the number of occurrences of rule $S \rightarrow aS$ in t.

We use below the following well-known relations $(0 < r < 1)$

$$\sum_{i=0}^{+\infty} r^i = \frac{1}{1-r}, \quad (31)$$

$$\sum_{i=1}^{+\infty} i \cdot r^{i-1} = \frac{1}{(1-r)^2}. \quad (32)$$

The derivational entropy of $\mathcal{G}_q$ can be directly computed from its definition as

$$\begin{aligned} H_d(p_{G,q}) &= -\sum_{i=0}^{+\infty} q^i \cdot (1-q) \cdot \log\left(q^i \cdot (1-q)\right) \\ &= -(1-q)\sum_{i=0}^{+\infty} q^i \log q^i + \\ &\quad -(1-q)\cdot\log(1-q)\cdot\sum_{i=0}^{+\infty} q^i \\ &= -(1-q)\cdot\log q \cdot \\ &\quad \sum_{i=0}^{+\infty} i \cdot q^i - \log(1-q) \\ &= -\frac{q}{1-q}\cdot\log q - \log(1-q). \quad (33) \end{aligned}$$

See Figure 1 for a plot of $H_d(p_{G,q})$ as a function of q.

If a tree bank is given, composed of occurrences of trees generated by G, the value of q can be estimated by applying the MLE or, equivalently, by minimizing the cross-entropy. We consider here several tree banks, to exemplify the behaviour of the cross-entropy depending on the structure of the sample of trees. The first tree bank $\mathcal{T}$ contains a single tree t with a single occurrence of rule $S \rightarrow aS$ and a single occurrence of rule $S \rightarrow a$. We then have $p_{\mathcal{T}}(t) = 1$ and $p_{G,q}(t) = q \cdot (1-q)$. The cross-entropy between distributions $p_{\mathcal{T}}$ and $p_{G,q}$ is then

$$\begin{aligned} H(p_{\mathcal{T}}, p_{G,q}) &= -\log q \cdot (1-q) \\ &= -\log q - \log(1-q). \quad (34) \end{aligned}$$

The cross-entropy $H(p_{\mathcal{T}}, p_{G,q})$, viewed as a function of q, is a convex-$\cup$ function and is plotted in Figure 1 (line indicated by $\frac{K}{d} = 1$, see below). We can obtain its minimum by finding a zero for the first derivative

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}q} H(p_{\mathcal{T}}, p_{G,q}) &= -\frac{1}{q} + \frac{1}{1-q} \\ &= \frac{2q-1}{q\cdot(1-q)} = 0, \quad (35) \end{aligned}$$

which gives $q = 0.5$. Note from Figure 1 that the minimum of $H(p_{\mathcal{T}}, p_{G,q})$ crosses the line corresponding to the derivational entropy, as should be expected from the result in Section 4.

More in general, for integers $d > 0$ and $K > 0$, consider a tree sample $\mathcal{T}_{d,K}$ consisting of d trees t_i, $1 \leq i \leq d$. Each t_i contains $k_i \geq 0$ occurrences of rule $S \rightarrow aS$ and one occurrence of rule $S \rightarrow a$. Thus we have $p_{\mathcal{T}_{d,K}}(t_i) = \frac{1}{d}$ and $p_{G,q}(t_i) = q^{k_i} \cdot (1-q)$. We let $\sum_{i=1}^{d} k_i = K$. The cross-entropy is

$$\begin{aligned} &H(p_{\mathcal{T}_{d,K}}, p_{G,q}) = \\ &\quad = -\sum_{i=0}^{d} \frac{1}{d} \cdot \log q^{k_i} - \log(1-q) \\ &\quad = -\frac{K}{d}\log q - \log(1-q). \quad (36) \end{aligned}$$

In Figure 1 we plot $H(p_{\mathcal{T}_{d,K}}, p_{G,q})$ in the case $\frac{K}{d} = 0.5$ and in the case $\frac{K}{d} = 1.5$. Again, we have that these curves intersect with the curve corresponding to the derivational entropy $H_d(p_{G,q})$ at the points were they take their minimum values.

7 Conclusions

We have shown in this paper that, when a PCFG is estimated from some tree distribution by minimizing the cross-entropy, then the cross-entropy takes the same value as the derivational entropy of the PCFG itself. As a special case, this result holds for the maximum likelihood estimator, widely applied in statistical natural language parsing. The result also holds for the relative weighted frequency estimator introduced in (Chi, 1999) as a generalization of the maximum likelihood estimator, and for the estimator introduced in (Nederhof, 2005) already discussed in the introduction. In a journal version of the present paper, which is under submission, we have also extended the results of Section 4 to the unsupervised estimation of a PCFG from a distribution defined over an infinite set of (unannotated) sentences and, as a particular case, to the well-knonw inside-outside algorithm (Manning and Schütze, 1999).

In practical applications, the results of Section 4 can be exploited in the computation of model tightness. In fact, cross-entropy indicates how much the estimated model fits the observed data, and is commonly exploited in comparison of different models on the same data set. We can then use the given relation between cross-entropy and derivational entropy to compute one of these two quantities from the other. For instance, in the case of the MLE method we can choose between the computation of the derivational entropy and the cross-entropy, depending basically on the instance of the problem at hand. As already mentioned, the computation of the derivational entropy requires cubic time in the number of nonterminals of the grammar. If this number is large, direct computation of (5) on the corpus might be more efficient. On the other hand, if the corpus at hand is very large, one might opt for direct computation of (3).

Acknowledgements

Helpful comments from Zhiyi Chi, Alberto lavelli, Mark-Jan Nederhof and Khalil Simaan are gratefully acknowledged.

References

T.L. Booth and R.A. Thompson. 1973. Applying probabilistic measures to abstract languages. *IEEE Transactions on Computers*, C-22(5):442–450, May.

E. Charniak. 2001. Immediate-head parsing for language models. In *39th Annual Meeting and 10th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference*, pages 116–123, Toulouse, France, July.

R. Chaudhuri, S. Pham, and O. N. Garcia. 1983. Solution of an open problem on probabilistic grammars. *IEEE Transactions on Computers*, 32(8):748–750.

C. Chelba and F. Jelinek. 1998. Exploiting syntactic structure for language modeling. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, volume 1, pages 225–231, Montreal, Quebec, Canada, August.

Z. Chi and S. Geman. 1998. Estimation of probabilistic context-free grammars. *Computational Linguistics*, 24(2):299–305.

Z. Chi. 1999. Statistical properties of probabilistic context-free grammars. *Computational Linguistics*, 25(1):131–160.

M. Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, pages 589–638.

J.E. Hopcroft and J.D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

C.D. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. Massachusetts Institute of Technology.

M.-J. Nederhof and G. Satta. 2004. Kullback-Leibler distance between probabilistic context-free grammars and probabilistic finite automata. In *Proc. of the 20^{th} COLING*, volume 1, pages 71–77, Geneva, Switzerland.

M.-J. Nederhof. 2005. A general technique to train language models on language models. *Computational Linguistics*, 31(2):173–185.

B. Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Y. Schabes. 1992. Stochastic lexicalized tree-adjoining grammars. In *Proc. of the fifteenth International Conference on Computational Linguistics*, volume 2, pages 426–432, Nantes, August.

Estimation of Consistent Probabilistic Context-free Grammars

Mark-Jan Nederhof
Max Planck Institute
for Psycholinguistics
P.O. Box 310
NL-6500 AH Nijmegen
The Netherlands
MarkJan.Nederhof@mpi.nl

Giorgio Satta
Dept. of Information Engineering
University of Padua
via Gradenigo, 6/A
I-35131 Padova
Italy
satta@dei.unipd.it

Abstract

We consider several empirical estimators for probabilistic context-free grammars, and show that the estimated grammars have the so-called consistency property, under the most general conditions. Our estimators include the widely applied expectation maximization method, used to estimate probabilistic context-free grammars on the basis of unannotated corpora. This solves a problem left open in the literature, since for this method the consistency property has been shown only under restrictive assumptions on the rules of the source grammar.

1 Introduction

Probabilistic context-free grammars are one of the most widely used formalisms in current work in statistical natural language parsing and stochastic language modeling. An important property for a probabilistic context-free grammar is that it be consistent, that is, the grammar should assign probability of one to the set of all finite strings or parse trees that it generates. In other words, the grammar should not lose probability mass with strings or trees of infinite length.

Several methods for the empirical estimation of probabilistic context-free grammars have been proposed in the literature, based on the optimization of some function on the probabilities of the observed data, such as the maximization of the likelihood of a tree bank or a corpus of unannotated sentences. It has been conjectured in (Wetherell, 1980) that these methods always provide probabilistic context-free grammars with the consistency property. A first result in this direction was presented in (Chaudhuri et al., 1983), by showing that a probabilistic context-free grammar estimated by maximizing the likelihood of a sample of parse trees is always consistent.

In later work by (Sánchez and Benedí, 1997) and (Chi and Geman, 1998), the result was independently extended to expectation maximization, which is an unsupervised method exploited to estimate probabilistic context-free grammars by finding local maxima of the likelihood of a sample of unannotated sentences. The proof in (Sánchez and Benedí, 1997) makes use of spectral analysis of expectation matrices, while the proof in (Chi and Geman, 1998) is based on a simpler counting argument. Both these proofs assume restrictions on the underlying context-free grammars. More specifically, in (Chi and Geman, 1998) empty rules and unary rules are not allowed, thus excluding infinite ambiguity, that is, the possibility that some string in the input sample has an infinite number of derivations in the grammar. The treatment of general form context-free grammars has been an open problem so far.

In this paper we consider several estimation methods for probabilistic context-free grammars, and we show that the resulting grammars have the consistency property. Our proofs are applicable under the most general conditions, and our results also include the expectation maximization method, thus solving the open problem discussed above. We use an alternative proof technique with respect to pre-

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 343–350,
New York, June 2006.

vious work, based on an already known renormalization construction for probabilistic context-free grammars, which has been used in the context of language modeling.

The structure of this paper is as follows. We provide some preliminary definitions in Section 2, followed in Section 3 by a brief overview of the estimation methods we investigate in this paper. In Section 4 we prove some properties of a renormalization technique for probabilistic context-free grammars, and use this property to show our main results in Section 5. Section 6 closes with some concluding remarks.

2 Preliminaries

In this paper we use mostly standard notation, as for instance in (Hopcroft and Ullman, 1979) and (Booth and Thompson, 1973), which we summarize below.

A **context-free grammar** (CFG) is a 4-tuple $G = (N, \Sigma, S, R)$ where N and Σ are finite disjoint sets of nonterminal and terminal symbols, respectively, $S \in N$ is the start symbol and R is a finite set of rules. Each rule has the form $A \rightarrow \alpha$, where $A \in N$ and $\alpha \in (\Sigma \cup N)^*$. We write V for set $\Sigma \cup N$.

Each CFG G is associated with a **left-most derive** relation $\Rightarrow$, defined on triples consisting of two strings $\gamma, \delta \in V^*$ and a rule $\pi \in R$. We write $\gamma \stackrel{\pi}{\Rightarrow} \delta$ if and only if $\gamma = uA\gamma'$ and $\delta = u\alpha\gamma'$, for some $u \in \Sigma^*$, $\gamma' \in V^*$, and $\pi = (A \rightarrow \alpha)$. A **left-most derivation** for G is a string $d = \pi_1 \cdots \pi_m$, $m \geq 0$, such that $\gamma_0 \stackrel{\pi_1}{\Rightarrow} \gamma_1 \stackrel{\pi_2}{\Rightarrow} \cdots \stackrel{\pi_m}{\Rightarrow} \gamma_m$, for some $\gamma_0, \ldots, \gamma_m \in V^*$; $d = \varepsilon$ (where ε denotes the empty string) is also a left-most derivation. In the remainder of this paper, we will let the term derivation always refer to left-most derivation. If $\gamma_0 \stackrel{\pi_1}{\Rightarrow} \cdots \stackrel{\pi_m}{\Rightarrow} \gamma_m$ for some $\gamma_0, \ldots, \gamma_m \in V^*$, then we say that $d = \pi_1 \cdots \pi_m$ **derives** γ_m from γ_0 and we write $\gamma_0 \stackrel{d}{\Rightarrow} \gamma_m$; $d = \varepsilon$ derives any $\gamma_0 \in V^*$ from itself.

A (left-most) derivation d such that $S \stackrel{d}{\Rightarrow} w$, $w \in \Sigma^*$, is called a **complete** derivation. If d is a complete derivation, we write $y(d)$ to denote the (unique) string $w \in \Sigma^*$ such that $S \stackrel{d}{\Rightarrow} w$. We define $D(G)$ to be the set of all complete derivations for G. The language generated by G is the set of all strings derived by complete derivations, i.e., $L(G) = \{y(d) \,|\, d \in D(G)\}$. It is well-known that there is a one-to-one correspondence between complete derivations and parse trees for strings in $L(G)$.

For $X \in V$ and $\alpha \in V^*$, we write $f(X, \alpha)$ to denote the number of occurrences of X in α. For $(A \rightarrow \alpha) \in R$ and a derivation d, $f(A \rightarrow \alpha, d)$ denotes the number of occurrences of $A \rightarrow \alpha$ in d. We let $f(A, d) = \sum_\alpha f(A \rightarrow \alpha, d)$.

A **probabilistic** CFG (PCFG) is a pair $\mathcal{G} = (G, p_G)$, where G is a CFG and p_G is a function from R to real numbers in the interval $[0, 1]$. We say that $\mathcal{G}$ is **proper** if, for every $A \in N$, we have

$$\sum_{A \rightarrow \alpha} p_G(A \rightarrow \alpha) \;=\; 1. \tag{1}$$

Function p_G can be used to assign probabilities to derivations of the underlying CFG G, in the following way. For $d = \pi_1 \cdots \pi_m \in R^*$, $m \geq 0$, we define

$$p_G(d) \;=\; \prod_{i=1}^{m} p_G(\pi_i). \tag{2}$$

Note that $p_G(\varepsilon) = 1$. The probability of a string $w \in \Sigma^*$ is defined as

$$p_G(w) \;=\; \sum_{y(d)=w} p_G(d). \tag{3}$$

A PCFG is **consistent** if

$$\sum_{w} p_G(w) \;=\; 1. \tag{4}$$

Consistency implies that the PCFG defines a probability distribution over both sets $D(G)$ and $L(G)$. If a PCFG is proper, then consistency means that no probability mass is lost in derivations of infinite length. All PCFGs in this paper are implicitly assumed to be proper, unless otherwise stated.

3 Estimation of PCFGs

In this section we give a brief overview of some estimation methods for PCFGs. These methods will be later investigated to show that they always provide consistent PCFGs.

In natural language processing applications, estimation of a PCFG is usually carried out on the basis of a tree bank, which in this paper we assume to be a **sample**, that is, a finite multiset, of complete derivations. Let $\mathcal{D}$ be such a sample, and let D be

the underlying set of derivations. For $d \in D$, we let $f(d, \mathcal{D})$ be the multiplicity of d in $\mathcal{D}$, that is, the number of occurrences of d in $\mathcal{D}$. We define

$$f(A \rightarrow \alpha, \mathcal{D}) = \sum_{d \in D} f(d, \mathcal{D}) \cdot f(A \rightarrow \alpha, d), \quad (5)$$

and let $f(A, \mathcal{D}) = \sum_{\alpha} f(A \rightarrow \alpha, \mathcal{D})$.

Consider a CFG $G = (N, \Sigma, R, S)$ defined by all and only the nonterminals, terminals and rules observed in D. The criterion of maximum likelihood estimation (MLE) prescribes the construction of a PCFG $\mathcal{G} = (G, p_G)$ such that p_G maximizes the likelihood of $\mathcal{D}$, defined as

$$p_G(\mathcal{D}) = \prod_{d \in D} p_G(d)^{f(d, \mathcal{D})}, \quad (6)$$

subject to the properness conditions $\sum_{\alpha} p_G(A \rightarrow \alpha) = 1$ for each $A \in N$. The maximization problem above has a unique solution, provided by the estimator (see for instance (Chi and Geman, 1998))

$$p_G(A \rightarrow \alpha) = \frac{f(A \rightarrow \alpha, \mathcal{D})}{f(A, \mathcal{D})}. \quad (7)$$

We refer to this as the supervised MLE method.

In applications in which a tree bank is not available, one might still use the MLE criterion to train a PCFG in an unsupervised way, on the basis of a sample of unannotated sentences, also called a corpus. Let us call $\mathcal{C}$ such a sample and C the underlying set of sentences. For $w \in C$, we let $f(w, \mathcal{C})$ be the multiplicity of w in $\mathcal{C}$.

Assume a CFG $G = (N, \Sigma, R, S)$ that is able to generate all of the sentences in C, and possibly more. The MLE criterion prescribes the construction of a PCFG $\mathcal{G} = (G, p_G)$ such that p_G maximizes the likelihood of $\mathcal{C}$, defined as

$$p_G(\mathcal{C}) = \prod_{w \in C} p_G(w)^{f(w, \mathcal{C})}, \quad (8)$$

subject to the properness conditions as in the supervised case above. The above maximization problem provides a system of $|R|$ nonlinear equations (see (Chi and Geman, 1998))

$$p_G(A \rightarrow \alpha) = \frac{\sum_{w \in C} f(w, \mathcal{C}) \cdot E_{p_G(d \,|\, w)} f(A \rightarrow \alpha, d)}{\sum_{w \in C} f(w, \mathcal{C}) \cdot E_{p_G(d \,|\, w)} f(A, d)}, \quad (9)$$

where E_p denotes an expectation computed under distribution p, and $p_G(d \,|\, w)$ is the probability of derivation d conditioned by sentence w (so that $p_G(d \,|\, w) > 0$ only if $y(d) = w$). The solution to the above system is not unique, because of the nonlinearity. Furthermore, each solution of (9) identifies a point where the curve in (8) has partial derivatives of zero, but this does not necessarily correspond to a local maximum, let alone an absolute maximum. (A point with partial derivatives of zero that is not a local maximum could be a local minimum or even a so-called saddle point.) In practice, this system is typically solved by means of an iterative algorithm called inside/outside (Charniak, 1993), which implements the expectation maximization (EM) method (Dempster et al., 1977). Starting with an initial function p_G that probabilistically extends G, a so-called growth transformation is computed, defined as

$$\overline{p}_G(A \rightarrow \alpha) = \frac{\sum_{w \in C} f(w, \mathcal{C}) \cdot \sum_{y(d)=w} \frac{p_G(d)}{p_G(w)} \cdot f(A \rightarrow \alpha, d)}{\sum_{w \in C} f(w, \mathcal{C}) \cdot \sum_{y(d)=w} \frac{p_G(d)}{p_G(w)} \cdot f(A, d)}. \quad (10)$$

Following (Baum, 1972), one can show that $\overline{p}_G(\mathcal{C}) \geq p_G(\mathcal{C})$. Thus, by iterating the growth transformation above, we are guaranteed to reach a local maximum for (8), or possibly a saddle point. We refer to this as the unsupervised MLE method.

We now discuss a third estimation method for PCFGs, which was proposed in (Corazza and Satta, 2006). This method can be viewed as a generalization of the supervised MLE method to probability distributions defined over infinite sets of complete derivations. Let D be an infinite set of complete derivations using nonterminal symbols in N, start symbol $S \in N$ and terminal symbols in Σ. We assume that the set of rules that are observed in D is drawn from some finite set R. Let p_D be a probability distribution defined over D, that is, a function from set D to interval $[0, 1]$ such that $\sum_{d \in D} p_D(d) = 1$.

Consider the CFG $G = (N, \Sigma, R, S)$. Note that $D \subseteq D(G)$. We wish to extend G to some PCFG $\mathcal{G} = (G, p_G)$ in such a way that p_D is approximated by p_G (viewed as a distribution over complete derivations) as well as possible according to some criterion. One possible criterion is minimization of

the **cross-entropy** between p_D and p_G, defined as the expectation, under distribution p_D, of the information of the derivations in D computed under distribution p_G, that is

$$\begin{aligned} H(p_D \,||\, p_G) &= E_{p_D} \log \frac{1}{p_G(d)} \\ &= -\sum_{d \in D} p_D(d) \cdot \log p_G(d). \quad (11) \end{aligned}$$

We thus want to assign to the parameters $p_G(A \rightarrow \alpha)$, $A \rightarrow \alpha \in R$, the values that minimize (11), subject to the conditions $\sum_\alpha p_G(A \rightarrow \alpha) = 1$ for each $A \in N$. Note that minimization of the cross-entropy above is equivalent to minimization of the Kullback-Leibler distance between p_D and p_G. Also note that the likelihood of an infinite set of derivations would always be zero and therefore cannot be considered here.

The solution to the above minimization problem provides the estimator

$$p_G(A \rightarrow \alpha) = \frac{E_{p_D}\ f(A \rightarrow \alpha, d)}{E_{p_D}\ f(A, d)}. \quad (12)$$

A proof of this result appears in (Corazza and Satta, 2006), and is briefly summarized in Appendix A, in order to make this paper self-contained. We call the above estimator the cross-entropy minimization method.

The cross-entropy minimization method can be viewed as a generalization of the supervised MLE method in (7), as shown in what follows. Let $\mathcal{D}$ and D be defined as for the supervised MLE method. We define a distribution over D as

$$p_{\mathcal{D}}(d) = \frac{f(d, \mathcal{D})}{|\mathcal{D}|}. \quad (13)$$

Distribution $p_{\mathcal{D}}$ is usually called the **empirical distribution** associated with $\mathcal{D}$. Applying the estimator in (12) to $p_{\mathcal{D}}$, we obtain

$$\begin{aligned} &p_G(A \rightarrow \alpha) = \\ &= \frac{\sum_{d \in D}\ p_{\mathcal{D}}(d) \cdot f(A \rightarrow \alpha, d)}{\sum_{d \in D}\ p_{\mathcal{D}}(d) \cdot f(A, d)} \\ &= \frac{\sum_{d \in D}\ \frac{f(d,\mathcal{D})}{|\mathcal{D}|} \cdot f(A \rightarrow \alpha, d)}{\sum_{d \in D}\ \frac{f(d,\mathcal{D})}{|\mathcal{D}|} \cdot f(A, d)} \\ &= \frac{\sum_{d \in D}\ f(d, \mathcal{D}) \cdot f(A \rightarrow \alpha, d)}{\sum_{d \in D}\ f(d, \mathcal{D}) \cdot f(A, d)}. \quad (14) \end{aligned}$$

This is the supervised MLE estimator in (7). This reminds us of the well-known fact that maximizing the likelihood of a (finite) sample through a PCFG distribution amounts to minimizing the cross-entropy between the empirical distribution of the sample and the PCFG distribution itself.

4 Renormalization

In this section we recall a renormalization technique for PCFGs that was used before in (Abney et al., 1999), (Chi, 1999) and (Nederhof and Satta, 2003) for different purposes, and is exploited in the next section to prove our main results. In the remainder of this section, we assume a fixed, not necessarily proper PCFG $\mathcal{G} = (G, p_G)$, with $G = (N, \Sigma, S, R)$.

We define the **renormalization** of $\mathcal{G}$ as the PCFG $\mathcal{R}(\mathcal{G}) = (G, p_{\mathcal{R}})$ with $p_{\mathcal{R}}$ specified by

$$\begin{aligned} &p_{\mathcal{R}}(A \rightarrow \alpha) = \\ &\quad p_G(A \rightarrow \alpha) \cdot \frac{\sum_{d,w}\ p_G(\alpha \stackrel{d}{\Rightarrow} w)}{\sum_{d,w}\ p_G(A \stackrel{d}{\Rightarrow} w)}. \quad (15) \end{aligned}$$

It is not difficult to see that $\mathcal{R}(\mathcal{G})$ is a proper PCFG. We now show an important property of $\mathcal{R}(\mathcal{G})$, discussed before in (Nederhof and Satta, 2003) in the context of so-called weighted context-free grammars.

Lemma 1 *For each derivation d with $A \stackrel{d}{\Rightarrow} w$, $A \in N$ and $w \in \Sigma^*$, we have*

$$p_{\mathcal{R}}(A \stackrel{d}{\Rightarrow} w) = \frac{p_G(A \stackrel{d}{\Rightarrow} w)}{\sum_{d', w'}\ p_G(A \stackrel{d'}{\Rightarrow} w')}. \quad (16)$$

Proof. The proof is by induction on the length of d, written $|d|$. If $|d| = 1$ we must have $d = (A \rightarrow w)$, and thus $p_{\mathcal{R}}(d) = p_{\mathcal{R}}(A \rightarrow w)$. In this case, the statement of the lemma directly follows from (15).

Assume now $|d| > 1$ and let $\pi = (A \rightarrow \alpha)$ be the first rule used in d. Note that there must be at least one nonterminal symbol in α. We can then write α as $u_0 A_1 u_1 A_2 \cdots u_{q-1} A_q u_q$, for $q \geq 1$, $A_i \in N$, $1 \leq i \leq q$, and $u_j \in \Sigma^*$, $0 \leq j \leq q$. In words, $A_1, \ldots, A_q$ are all of the occurrences of nonterminals in α, as they appear from left to right. Consequently, we can write d in the form $d = \pi \cdot d_1 \cdots d_q$ for some derivations d_i, $1 \leq i \leq q$, with $A_i \stackrel{d_i}{\Rightarrow} w_i$, $|d_i| \geq 1$ and with

$w = u_0 w_1 u_1 w_2 \cdots u_{q-1} w_q u_q$. Below we use the fact that $p_{\mathcal{R}}(u_j \overset{\varepsilon}{\Rightarrow} u_j) = p_G(u_j \overset{\varepsilon}{\Rightarrow} u_j) = 1$ for each j with $0 \leq j \leq q$, and further using the definition of $p_{\mathcal{R}}$ and the inductive hypothesis, we can write

$$
\begin{aligned}
& p_{\mathcal{R}}(A \overset{d}{\Rightarrow} w) = \\
& = p_{\mathcal{R}}(A \to \alpha) \cdot \prod_{i=1}^{q} p_{\mathcal{R}}(A_i \overset{d_i}{\Rightarrow} w_i) \\
& = p_G(A \to \alpha) \cdot \frac{\sum_{d',w'} p_G(\alpha \overset{d'}{\Rightarrow} w')}{\sum_{d',w'} p_G(A \overset{d'}{\Rightarrow} w')} \cdot \\
& \quad \cdot \prod_{i=1}^{q} p_{\mathcal{R}}(A_i \overset{d_i}{\Rightarrow} w_i) \\
& = p_G(A \to \alpha) \cdot \frac{\sum_{d',w'} p_G(\alpha \overset{d'}{\Rightarrow} w')}{\sum_{d',w'} p_G(A \overset{d'}{\Rightarrow} w')} \cdot \\
& \quad \cdot \prod_{i=1}^{q} \frac{p_G(A_i \overset{d_i}{\Rightarrow} w_i)}{\sum_{d',w'} p_G(A_i \overset{d'}{\Rightarrow} w')} \\
& = p_G(A \to \alpha) \cdot \frac{\sum_{d',w'} p_G(\alpha \overset{d'}{\Rightarrow} w')}{\sum_{d',w'} p_G(A \overset{d'}{\Rightarrow} w')} \cdot \\
& \quad \cdot \frac{\prod_{i=1}^{q} p_G(A_i \overset{d_i}{\Rightarrow} w_i)}{\prod_{i=1}^{q} \sum_{d',w'} p_G(A_i \overset{d'}{\Rightarrow} w')} \\
& = p_G(A \to \alpha) \cdot \frac{\sum_{d',w'} p_G(\alpha \overset{d'}{\Rightarrow} w')}{\sum_{d',w'} p_G(A \overset{d'}{\Rightarrow} w')} \cdot \\
& \quad \cdot \frac{\prod_{i=1}^{q} p_G(A_i \overset{d_i}{\Rightarrow} w_i)}{\sum_{d',w'} p_G(\alpha \overset{d'}{\Rightarrow} w')} \\
& = p_G(A \to \alpha) \cdot \frac{\prod_{i=1}^{q} p_G(A_i \overset{d_i}{\Rightarrow} w_i)}{\sum_{d',w'} p_G(A \overset{d'}{\Rightarrow} w')} \cdot \\
& = \frac{p_G(A \overset{d}{\Rightarrow} w)}{\sum_{d',w'} p_G(A \overset{d'}{\Rightarrow} w')}. \qquad (17)
\end{aligned}
$$

■

As an easy corollary of Lemma 1, we have that $\mathcal{R}(\mathcal{G})$ is a consistent PCFG, as we can write

$$
\begin{aligned}
& \sum_{d,w} p_{\mathcal{R}}(S \overset{d}{\Rightarrow} w) = \\
& = \sum_{d,w} \frac{p_G(S \overset{d}{\Rightarrow} w)}{\sum_{d',w'} p_G(S \overset{d'}{\Rightarrow} w')} \\
& = \frac{\sum_{d,w} p_G(S \overset{d}{\Rightarrow} w)}{\sum_{d',w'} p_G(S \overset{d'}{\Rightarrow} w')} = 1. \qquad (18)
\end{aligned}
$$

5 Consistency

In this section we prove the main results of this paper, namely that all of the estimation methods discussed in Section 3 always provide consistent PCFGs. We start with a technical lemma, central to our results, showing that a PCFG that minimizes the cross-entropy with a distribution over any set of derivations must be consistent.

Lemma 2 *Let $\mathcal{G} = (G, p_G)$ be a proper PCFG and let p_D be a probability distribution defined over some set $D \subseteq D(G)$. If $\mathcal{G}$ minimizes function $H(p_D \,||\, p_G)$, then $\mathcal{G}$ is consistent.*

Proof. Let $G = (N, \Sigma, S, R)$, and assume that $\mathcal{G}$ is not consistent. We establish a contradiction. Since $\mathcal{G}$ is not consistent, we must have $\sum_{d,w} p_G(S \overset{d}{\Rightarrow} w) < 1$. Let then $\mathcal{R}(\mathcal{G}) = (G, p_{\mathcal{R}})$ be the renormalization of $\mathcal{G}$, defined as in (15). For any derivation $S \overset{d}{\Rightarrow} w$, $w \in \Sigma^*$, with d in D, we can use Lemma 1 and write

$$
\begin{aligned}
& p_{\mathcal{R}}(S \overset{d}{\Rightarrow} w) = \\
& = \frac{1}{\sum_{d',w'} p_G(S \overset{d'}{\Rightarrow} w')} \cdot p_G(S \overset{d}{\Rightarrow} w) \\
& > p_G(S \overset{d}{\Rightarrow} w). \qquad (19)
\end{aligned}
$$

In words, every complete derivation d in D has a probability in $\mathcal{R}(\mathcal{G})$ that is strictly greater than in $\mathcal{G}$. But this means $H(p_D \,||\, p_{\mathcal{R}}) < H(p_D \,||\, p_G)$, against our hypothesis. Therefore, $\mathcal{G}$ is consistent and p_G is a probability distribution over set $D(G)$. Thus function $H(p_D \,||\, p_G)$ can be interpreted as the cross-entropy. (Observe that in the statement of the lemma we have avoided the term 'cross-entropy', since cross-entropies are only defined for probability distributions.) ■

Lemma 2 directly implies that the cross-entropy minimization method in (12) always provides a consistent PCFG, since it minimizes cross-entropy for a distribution defined over a subset of $D(G)$. We have already seen in Section 3 that the supervised MLE method is a special case of the cross-entropy minimization method. Thus we can also conclude that a PCFG trained with the supervised MLE method is

always consistent. This provides an alternative proof of a property that was first shown in (Chaudhuri et al., 1983), as discussed in Section 1.

We now prove the same result for the unsupervised MLE method, without any restrictive assumption on the rules of our CFGs. This solves a problem that was left open in the literature (Chi and Geman, 1998); see again Section 1 for discussion. Let $\mathcal{C}$ and C be defined as in Section 3. We define the empirical distribution of $\mathcal{C}$ as

$$p_C(w) \;=\; \frac{f(w,\mathcal{C})}{|\mathcal{C}|}. \qquad (20)$$

Let $G = (N, \Sigma, S, R)$ be a CFG such that $C \subseteq L(G)$. Let $D(C)$ be the set of all complete derivations for G that generate sentences in C, that is, $D(C) = \{d \,|\, d \in D(G),\ y(d) \in C\}$.

Further, assume some probabilistic extension $\mathcal{G} = (G, p_G)$ of G, such that $p_G(d) > 0$ for every $d \in D(C)$. We define a distribution over $D(C)$ by

$$p_{D(C)}(d) \;=\; p_C(y(d)) \cdot \frac{p_G(d)}{p_G(y(d))}. \qquad (21)$$

It is not difficult to verify that

$$\sum_{d \in D(C)} p_{D(C)}(d) \;=\; 1. \qquad (22)$$

We now apply to $\mathcal{G}$ the estimator in (12), in order to obtain a new PCFG $\hat{\mathcal{G}} = (G, \hat{p}_G)$ that minimizes the cross-entropy between $p_{D(C)}$ and $\hat{p}_G$. According to Lemma 2, we have that $\hat{\mathcal{G}}$ is a consistent PCFG. Distribution $\hat{p}_G$ is specified by

$$\begin{aligned}
&\hat{p}_G(A \to \alpha) \;= \\
&\quad= \frac{\sum_{d \in D(C)} p_{D(C)}(d) \cdot f(A \to \alpha, d)}{\sum_{d \in D(C)} p_{D(C)}(d) \cdot f(A, d)} \\
&\quad= \frac{\sum_{d \in D(C)} \frac{f(y(d),\mathcal{C})}{|\mathcal{C}|} \cdot \frac{p_G(d)}{p_G(y(d))} \cdot f(A \to \alpha, d)}{\sum_{d \in D(C)} \frac{f(y(d),\mathcal{C})}{|\mathcal{C}|} \cdot \frac{p_G(d)}{p_G(y(d))} \cdot f(A, d)} \\
&\quad= \frac{\sum_{w \in C} f(w,\mathcal{C}) \cdot \sum_{y(d)=w} \frac{p_G(d)}{p_G(w)} \cdot f(A \to \alpha, d)}{\sum_{w \in C} f(w,\mathcal{C}) \cdot \sum_{y(d)=w} \frac{p_G(d)}{p_G(w)} \cdot f(A, d)} \\
&\quad= \frac{\sum_{w \in C} f(w,\mathcal{C}) \cdot E_{p_G(d\,|\,w)} f(A \to \alpha, d)}{\sum_{w \in C} f(w,\mathcal{C}) \cdot E_{p_G(d\,|\,w)} f(A, d)}. \qquad (23)
\end{aligned}$$

Since distribution p_G was arbitrarily chosen, subject to the only restriction that $p_G(d) > 0$ for every $d \in D(C)$, we have that (23) is the growth estimator (10) already discussed in Section 3. In fact, for each $w \in L(G)$ and $d \in D(G)$, we have $p_G(d \,|\, w) = \frac{p_G(d)}{p_G(w)}$. We conclude with the desired result, namely that a general form PCFG obtained at any iteration of the EM method for the unsupervised MLE is always consistent.

6 Conclusions

In this paper we have investigated a number of methods for the empirical estimation of probabilistic context-free grammars, and have shown that the resulting grammars have the so-called consistency property. This property guarantees that all the probability mass of the grammar is used for the finite strings it derives. Thus if the grammar is used in combination with other probabilistic models, as for instance in a speech processing system, consistency allows us to combine or compare scores from different modules in a sound way.

To obtain our results, we have used a novel proof technique that exploits an already known construction for the renormalization of probabilistic context-free grammars. Our proof technique seems more intuitive than arguments previously used in the literature to prove the consistency property, based on counting arguments or on spectral analysis. It is not difficult to see that our proof technique can also be used with probabilistic rewriting formalisms whose underlying derivations can be characterized by means of context-free rewriting. This is for instance the case with probabilistic tree-adjoining grammars (Schabes, 1992; Sarkar, 1998), for which consistency results have not yet been shown in the literature.

A Cross-entropy minimization

In order to make this paper self-contained, we sketch a proof of the claim in Section 3 that the estimator in (12) minimizes the cross entropy in (11). A full proof appears in (Corazza and Satta, 2006).

Let D, p_D and $G = (N, \Sigma, R, S)$ be defined as in Section 3. We want to find a proper PCFG $\mathcal{G} = (G, p_G)$ such that the cross-entropy $H(p_D \,||\, p_G)$ is minimal. We use Lagrange multipliers λ_A for each $A \in N$ and define the form

$$\nabla \;=\; \sum_{A \in N} \lambda_A \cdot \Big(\sum_{\alpha} p_G(A \to \alpha) - 1\Big) \;+$$

$$- \sum_{d \in D} p_D(d) \cdot \log p_G(d). \quad (24)$$

We now consider all the partial derivatives of ∇. For each $A \in N$ we have

$$\frac{\partial \nabla}{\partial \lambda_A} = \sum_{\alpha} p_G(A \rightarrow \alpha) - 1. \quad (25)$$

For each $(A \rightarrow \alpha) \in R$ we have

$$\begin{aligned}
&\frac{\partial \nabla}{\partial p_G(A \rightarrow \alpha)} = \\
&= \lambda_A - \frac{\partial}{\partial p_G(A \rightarrow \alpha)} \sum_{d \in D} p_D(d) \cdot \log p_G(d) \\
&= \lambda_A - \sum_{d \in D} p_D(d) \cdot \frac{\partial}{\partial p_G(A \rightarrow \alpha)} \log p_G(d) \\
&= \lambda_A - \sum_{d \in D} p_D(d) \cdot \frac{\partial}{\partial p_G(A \rightarrow \alpha)} \\
&\quad \log \prod_{(B \rightarrow \beta) \in R} p_G(B \rightarrow \beta)^{f(B \rightarrow \beta, d)} \\
&= \lambda_A - \sum_{d \in D} p_D(d) \cdot \frac{\partial}{\partial p_G(A \rightarrow \alpha)} \\
&\quad \sum_{(B \rightarrow \beta) \in R} f(B \rightarrow \beta, d) \cdot \log p_G(B \rightarrow \beta) \\
&= \lambda_A - \sum_{d \in D} p_D(d) \cdot \sum_{(B \rightarrow \beta) \in R} f(B \rightarrow \beta, d) \cdot \\
&\quad \frac{\partial}{\partial p_G(A \rightarrow \alpha)} \log p_G(B \rightarrow \beta) \\
&= \lambda_A - \sum_{d \in D} p_D(d) \cdot f(A \rightarrow \alpha, d) \cdot \\
&\quad \cdot \frac{1}{\ln(2)} \cdot \frac{1}{p_G(A \rightarrow \alpha)} \\
&= \lambda_A - \frac{1}{\ln(2)} \cdot \frac{1}{p_G(A \rightarrow \alpha)} \cdot \\
&\quad \cdot \sum_{d \in D} p_D(d) \cdot f(A \rightarrow \alpha, d) \\
&= \lambda_A - \frac{1}{\ln(2)} \cdot \frac{1}{p_G(A \rightarrow \alpha)} \cdot \\
&\quad \cdot E_{p_D} f(A \rightarrow \alpha, d). \quad (26)
\end{aligned}$$

By setting to zero all of the above partial derivatives, we obtain a system of $|N|+|R|$ equations, which we must solve. From $\frac{\partial \nabla}{\partial p_G(A \rightarrow \alpha)} = 0$ we obtain

$$\lambda_A \cdot \ln(2) \cdot p_G(A \rightarrow \alpha) = E_{p_D} f(A \rightarrow \alpha, d). \quad (27)$$

We sum over all strings α such that $(A \rightarrow \alpha) \in R$, deriving

$$\begin{aligned}
&\lambda_A \cdot \ln(2) \cdot \sum_{\alpha} p_G(A \rightarrow \alpha) = \\
&= \sum_{\alpha} E_{p_D} f(A \rightarrow \alpha, d) \\
&= \sum_{\alpha} \sum_{d \in D} p_D(d) \cdot f(A \rightarrow \alpha, d) \\
&= \sum_{d \in D} p_D(d) \cdot \sum_{\alpha} f(A \rightarrow \alpha, d) \\
&= \sum_{d \in D} p_D(d) \cdot f(A, d) \\
&= E_{p_D} f(A, d). \quad (28)
\end{aligned}$$

From each equation $\frac{\partial \nabla}{\partial \lambda_A} = 0$ we obtain $\sum_{\alpha} p_G(A \rightarrow \alpha) = 1$ for each $A \in N$ (our original constraints). Combining this with (28) we obtain

$$\lambda_A \cdot \ln(2) = E_{p_D} f(A, d). \quad (29)$$

Replacing (29) into (27) we obtain, for every rule $(A \rightarrow \alpha) \in R$,

$$p_G(A \rightarrow \alpha) = \frac{E_{p_D} f(A \rightarrow \alpha, d)}{E_{p_D} f(A, d)}. \quad (30)$$

This is the estimator introduced in Section 3.

References

S. Abney, D. McAllester, and F. Pereira. 1999. Relating probabilistic grammars and automata. In *37th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 542–549, Maryland, USA, June.

L. E. Baum. 1972. An inequality and associated maximization technique in statistical estimations of probabilistic functions of Markov processes. *Inequalities*, 3:1–8.

T.L. Booth and R.A. Thompson. 1973. Applying probabilistic measures to abstract languages. *IEEE Transactions on Computers*, C-22(5):442–450, May.

E. Charniak. 1993. *Statistical Language Learning*. MIT Press.

R. Chaudhuri, S. Pham, and O. N. Garcia. 1983. Solution of an open problem on probabilistic grammars. *IEEE Transactions on Computers*, 32(8):748–750.

Z. Chi and S. Geman. 1998. Estimation of probabilistic context-free grammars. *Computational Linguistics*, 24(2):299–305.

Z. Chi. 1999. Statistical properties of probabilistic context-free grammars. *Computational Linguistics*, 25(1):131–160.

A. Corazza and G. Satta. 2006. Cross-entropy and estimation of probabilistic context-free grammars. In *Proc. of HLT/NAACL 2006 Conference (this volume)*, New York.

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

J.E. Hopcroft and J.D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

M.-J. Nederhof and G. Satta. 2003. Probabilistic parsing as intersection. In *8th International Workshop on Parsing Technologies*, pages 137–148, LORIA, Nancy, France, April.

J.-A. Sánchez and J.-M. Benedí. 1997. Consistency of stochastic context-free grammars from probabilistic estimation based on growth transformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9):1052–1055, September.

A. Sarkar. 1998. Conditions on consistency of probabilistic tree adjoining grammars. In *Proc. of the 36^{th} ACL*, pages 1164–1170, Montreal, Canada.

Y. Schabes. 1992. Stochastic lexicalized tree-adjoining grammars. In *Proc. of the 14^{th} COLING*, pages 426–432, Nantes, France.

C. S. Wetherell. 1980. Probabilistic languages: A review and some open questions. *Computing Surveys*, 12(4):361–379.

A Better N-Best List: Practical Determinization of Weighted Finite Tree Automata

Jonathan May
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
jonmay@isi.edu

Kevin Knight
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
knight@isi.edu

Abstract

Ranked lists of output trees from syntactic statistical NLP applications frequently contain multiple repeated entries. This redundancy leads to misrepresentation of tree weight and reduced information for debugging and tuning purposes. It is chiefly due to nondeterminism in the weighted automata that produce the results. We introduce an algorithm that determinizes such automata while preserving proper weights, returning the sum of the weight of all multiply derived trees. We also demonstrate our algorithm's effectiveness on two large-scale tasks.

1 Introduction

A useful tool in natural language processing tasks such as translation, speech recognition, parsing, etc., is the ranked list of results. Modern systems typically produce competing partial results internally and return only the top-scoring complete result to the user. They are, however, also capable of producing lists of runners-up, and such lists have many practical uses:

- The lists may be inspected to determine the quality of runners-up and motivate model changes.
- The lists may be re-ranked with extra knowledge sources that are difficult to apply during the main search.
- The lists may be used with annotation and a tuning process, such as in (Collins and Roark, 2004), to iteratively alter feature weights and improve results.

Figure 1 shows the best 10 English translation parse trees obtained from a syntax-based translation system based on (Galley, et. al., 2004). Notice that the same tree occurs multiple times in this list. This repetition is quite characteristic of the output of ranked lists. It occurs because many systems, such as the ones proposed by (Bod, 1992), (Galley, et. al., 2004), and (Langkilde and Knight, 1998) represent their result space in terms of weighted partial results of various sizes that may be assembled in multiple ways. There is in general more than one way to assemble the partial results to derive the same complete result. Thus, the n-best list of results is really an n-best list of *derivations*.

When list-based tasks, such as the ones mentioned above, take as input the top n results for some constant n, the effect of repetition on these tasks is deleterious. A list with many repetitions suffers from a lack of useful information, hampering diagnostics. Repeated results prevent alternatives that would be highly ranked in a secondary reranking system from even being considered. And a list of fewer unique trees than expected can cause overfitting when this list is used to tune. Furthermore, the actual weight of obtaining any particular tree is split among its repetitions, distorting the actual relative weights between trees.

(Mohri, 1997) encountered this problem in speech recognition, and presented a solution to the problem of repetition in n-best lists of strings that are derived from finite-state automata. That work described a way to use a powerset construction along

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 351–358,
New York, June 2006.

```
34.73:  S(NP-C(NPB(DT(this) NNS(cases))) VP(VBD(had) VP-C(VBN(caused) NP-C(NPB(DT(the) JJ(american) NNS(protests))))) .(.))
•34.74:  S(NP-C(NPB(DT(this) NNS(cases))) VP(VBD(had) VP-C(VBN(aroused) NP-C(NPB(DT(the) JJ(american) NNS(protests))))) .(.))
34.83:  S(NP-C(NPB(DT(this) NNS(cases))) VP(VBD(had) VP-C(VBN(caused) NP-C(NPB(DT(the) JJ(american) NNS(protests))))) .(.))
•34.83:  S(NP-C(NPB(DT(this) NNS(cases))) VP(VBD(had) VP-C(VBN(aroused) NP-C(NPB(DT(the) JJ(american) NNS(protests))))) .(.))
34.84:  S(NP-C(NPB(DT(this) NNS(cases))) VP(VBD(had) VP-C(VBN(caused) NP-C(NPB(DT(the) JJ(american) NNS(protests))))) .(.))
34.85:  S(NP-C(NPB(DT(this) NNS(cases))) VP(VBD(had) VP-C(VBN(caused) NP-C(NPB(DT(the) JJ(american) NNS(protests))))) .(.))
•34.85:  S(NP-C(NPB(DT(this) NNS(cases))) VP(VBD(had) VP-C(VBN(aroused) NP-C(NPB(DT(the) JJ(american) NNS(protests))))) .(.))
•34.85:  S(NP-C(NPB(DT(this) NNS(cases))) VP(VBD(had) VP-C(VBN(aroused) NP-C(NPB(DT(the) JJ(american) NNS(protests))))) .(.))
34.87:  S(NP-C(NPB(DT(this) NNS(cases))) VP(VBD(had) VP-C(VB(arouse) NP-C(NPB(DT(the) JJ(american) NNS(protests))))) .(.))
•34.92:  S(NP-C(NPB(DT(this) NNS(cases))) VP(VBD(had) VP-C(VBN(aroused) NP-C(NPB(DT(the) JJ(american) NNS(protests))))) .(.))
```

Figure 1: Ranked list of machine translation results with repeated trees. Scores shown are negative logs of calculated weights, thus a lower score indicates a higher weight. The bulleted sentences indicate identical trees.

with an innovative bookkeeping system to determinize the automaton, resulting in an automaton that preserves the language but provides a single, properly weighted derivation for each string in it. Put another way, if the input automaton has the ability to generate the same string with different weights, the output automaton generates that string with weight equal to the sum of all of the generations of that string in the input automaton. In (Mohri and Riley, 2002) this technique was combined with a procedure for efficiently obtaining n-best ranked lists, yielding a list of string results with no repetition.

In this paper we extend that work to deal with grammars that produce trees. *Regular tree grammars* (Brainerd, 1969), which subsume the *tree substitution grammars* developed in the NLP community (Schabes, 1990), are of particular interest to those wishing to work with additional levels of structure that string grammars cannot provide. The application to parsing is natural, and in machine translation tree grammars can be used to model syntactic transfer, control of function words, reordering, and target-language well-formedness. In the world of automata these grammars have as a natural dual the *finite tree recognizer* (Doner, 1970). Like tree grammars and packed forests, they are compact ways of representing very large sets of trees. We will present an algorithm for determinizing weighted finite tree recognizers, and use a variant of the procedure found in (Huang and Chiang, 2005) to obtain n-best lists of trees that are weighted correctly and contain no repetition.

Section 2 describes related work. In Section 3, we introduce the formalisms of tree automata, specifically the tree-to-weight transducer. In Section 4, we present the algorithm. Finally, in Section 5 we show the results of applying weighted determinization to recognizers obtained from the packed forest output of two natural language tasks.

2 Previous Work

The formalisms of tree automata are summarized well in (Gecseg and Steinby, 1984). Bottom-up tree recognizers are due to (Thatcher and Wright, 1968), (Doner, 1970), and (Magidor and Moran, 1969). Top-down tree recognizers are due to (Rabin, 1969) and (Magidor and Moran, 1969). (Comon, et. al., 1997) show the determinization of unweighted finite-state tree automata, and prove its correctness. (Borchardt and Vogler, 2003) present determinization of weighted finite-state tree automata with a different method than the one we present here. While our method is applicable to finite tree sets, the previous method claims the ability to determinize some classes of infinite tree sets. However, for the finite case the previous method produces an automaton with size on the order of the number of derivations, so the technique is limited when applied to real world data.

3 Grammars, Recognizers, and Transducers

As described in (Gecseg and Steinby, 1984), tree automata may be broken into two classes, recognizers and transducers. Recognizers read tree input and decide whether the input is in the language represented by the recognizer. Formally, a bottom-up tree recognizer R is defined by $R = (Q, \Sigma, i, F, E)$:[1]

- Q is a finite set of states,

[1] Readers familiar with (Gecseg and Steinby, 1984) will notice that we have introduced a start state, modified the notion of initial assignment, and changed the arity of nullary symbols to unary symbols. This is to make tree automata more palatable to those accustomed to string automata and to allow for a useful graphical interpretation.

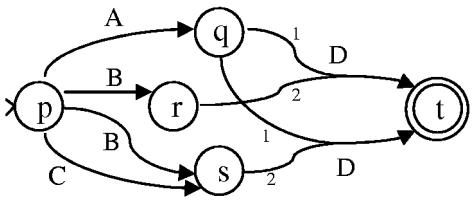


Figure 2: Visualization of a bottom-up tree recognizer

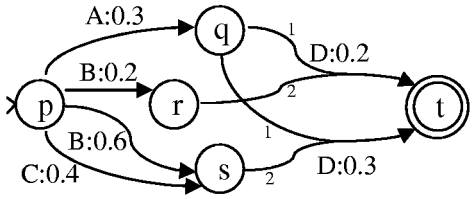


Figure 3: Bottom-up tree-to-weight transducer

- Σ is a ranked alphabet,
- $i \in Q$ is the initial state,
- $F \subseteq Q$ is a set of final states, and
- $E \subseteq \overbrace{Q \times ... \times Q}^{k} \times \Sigma^{(k)} \times Q$ is a finite set of transitions from a vector of k states to one state that reads a k-ary symbol.

Consider the following tree recognizer:

- $Q = \{p, q, r, s, t\}$
- $\Sigma = \{D/2, A/1, B/1, C/1\}$[2]
- $i = p$
- $F = \{t\}$
- $E = \{(\langle p \rangle, A, q), (\langle p \rangle, B, r), (\langle p \rangle, B, s), (\langle p \rangle, C, s), (\langle q, r \rangle, D, t), (\langle q, s \rangle, D, t)\}$

As with string automata, it is helpful to have a visualization to understand what the recognizer is recognizing. Figure 2 provides a visualization of the recognizer above. Notice that some members of E are drawn as arcs with multiple (and ordered) tails. This is the key difference in visualization between string and tree automata – to capture the arity of the symbol being read we must visualize the automata as an ordered hypergraph.

The function of the members of E in the hypergraph visualization leads us to refer to the vector of k states as an *input vector* of states, and the single state as an *output state*. We will refer to E as the *transition set* of the recognizer.

In string automata, a *path* through a recognizer consists of a sequence of edges that can be followed from a start to an end state. The concatenation of labels of the edges of a path, typically in a left-to-right order, forms a string in the recognizer's language. In tree automata, however, a *hyperpath* through a recognizer consists of a sequence of hyperedges that can be followed, sometimes in parallel, from a start to an end state. We arrange the labels of the hyperedges to form a tree in the recognizer's language but must now consider proper order in two dimensions. The proper vertical order is specified by the order of application of transitions, i.e., the labels of transitions followed earlier are placed lower in the tree than the labels of transitions followed later. The proper horizontal order within one level of the tree is specified by the order of states in a transition's input vector. In the example recognizer, the trees $\overset{D}{\widehat{A\ \ B}}$ and $\overset{D}{\widehat{A\ \ C}}$ are valid. Notice that $\overset{D}{\widehat{A\ \ B}}$ may be recognized in two different hyperpaths.

Like tree recognizers, tree transducers read tree input and decide whether the input is in the language, but they simultaneously produce some output as well. Since we wish to associate a weight with every acceptable tree in a language, we will consider transducers that produce weights as their output. Note that in transitioning from recognizers to transducers we are following the convention established in (Mohri, 1997) where a transducer with weight outputs is used to represent a weighted recognizer. One may consider the determinization of tree-to-weight transducers as equivalent to the determinization of weighted tree recognizers.

Formally, a bottom-up tree-to-weight transducer T is defined by $T = (Q, \Sigma, i, F, E, \lambda, \rho)$ where Q, Σ, i, and F are defined as for recognizers, and:

- $E \subseteq \overbrace{Q \times ... \times Q}^{k} \times \Sigma^{(k)} \times Q \times \mathcal{R}_+$ is a finite set of transitions from a vector of k states to one state, reading a k-ary symbol and outputting some weight
- λ is the initial weight function mapping i to $\mathcal{R}_+$
- ρ is the final weight function mapping F

[2]The number denotes the arity of the symbol.

to $\mathcal{R}_+$.

We must also specify a convention for propagating the weight calculated in every transition. This can be explicitly defined for each transition but we will simplify matters by defining the propagation of the weight to a destination state as the multiplication of the weight at each source state with the weight of the production.

We modify the previous example by adding weights as follows: As an example, consider the following tree-to-weight transducer (Q, Σ, i, and F are as before):

- $E = \{(\langle p\rangle, A, q, .3), (\langle p\rangle, B, r, .2), (\langle p\rangle, B, s, .6), (\langle p\rangle, C, s, .4), (\langle q, r\rangle, D, t, .2), (\langle q, s\rangle, D, t, .3)\}$
- $\lambda = \rho = 1$

Figure 3 shows the addition of weights onto the automata, forming the above transducer. Notice the tree $\overset{D}{\underset{A\quad C}{\frown}}$ yields the weight 0.036 ($0.3 \times 0.3 \times 0.4$), and $\overset{D}{\underset{A\quad B}{\frown}}$ yields the weight 0.012 ($0.2 \times 0.3 \times 0.2$) or 0.054 ($0.3 \times 0.3 \times 0.6$), depending on the hyperpath followed.

This transducer is an example of a *nonsubsequential* transducer. A tree transducer is *subsequential* if for each vector $\mathbf{v}$ of k states and each $x \in \Sigma^{(k)}$ there is at most one transition in E with input vector $\mathbf{v}$ and label x. These restrictions ensure a subsequential transducer yields a single output for each possible input, that is, it is deterministic in its output.

Because we will reason about the destination state of a transducer transition and the weight of a transducer transition separately, we make the following definition. For a given $e = (\mathbf{v}, x, q, w) \in E$ where $\mathbf{v}$ is a vector of k states, $x \in \Sigma^{(k)}$, $q \in Q$, and $w \in \mathcal{R}_+$, let $\delta(\mathbf{v}, x) = q$ and $\sigma(\mathbf{v}, x) = w$. Equivalent shorthand forms are $\delta(e)$ and $\sigma(e)$.

4 Determinization

The determinization algorithm is presented as Algorithm 1. It takes as input a bottom-up tree-to-weight transducer τ_1 and returns as output a subsequential bottom-up tree-to-weight transducer τ_2 such that the tree language recognized by τ_2 is equivalent to that of τ_1 and the output weight given input tree t on τ_2 is equal to the sum of all possible output weights given t on τ_1. Like the algorithm of (Mohri, 1997), this algorithm will terminate for automata that recognize finite tree languages. It may terminate on some automata that recognize infinite tree languages, but we do not consider any of these cases in this work.

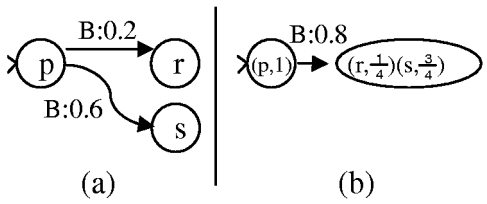


Figure 4: a) Portion of a transducer before determinization; b) The same portion after determinization

Determinizing a tree-to-weight transducer can be thought of as a two-stage process. First, the structure of the automata must be determined such that a single hyperpath exists for each recognized input tree. This is achieved by a classic powerset construction, i.e., a state must be constructed in the output transducer that represents all the possible reachable destination states given an input and a label. Because we are working with tree automata, our input is a vector of states, not a single state. A comparable powerset construction on unweighted tree automata and a proof of correctness can be found in (Comon, et. al., 1997).

The second consideration to weighted determinization is proper propagation of weights. For this we will use (Mohri, 1997)'s concept of the *residual weight*. We represent in the construction of states in the output transducer not only a subset of states of the input transducer, but also a number associated with each of these states, called the residual. Since we want τ_2's hyperpath of a particular input tree to have as its associated weight the sum of the weights of the all of τ_1's hyperpaths of the input tree, we replace a set of hyperedges in τ_1 that have the same input state vector and label with a single hyperedge in τ_2 bearing the label and the sum of τ_1's hyperedge weights. The destination state of the τ_2 hyperedge represents the states reachable by τ_1's applicable hyperedges and for each state, the proportion of the weight from the relevant τ_1 transition.

Figure 4 shows the determinization of a portion of the example transducer. Note that the hyperedge

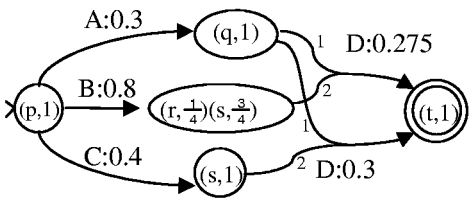


Figure 5: Determinized bottom-up tree-to-weight transducer

leading to state r in the input transducer contributes $\frac{1}{4}$ of the weight on the output transducer hyperedge and the hyperedge leading to state s in the input transducer contributes the remaining $\frac{3}{4}$. This is reflected in the state construction in the output transducer. The complete determinization of the example transducer is shown in Figure 5.

To encapsulate the representation of states from the input transducer and associated residual weights, we define a state in the output transducer as a set of (q, w) tuples, where $q \in Q_1$ and $w \in \mathcal{R}_+$. Since the algorithm builds new states progressively, we will need to represent a vector of states from the output transducer, typically depicted as $\mathbf{v}$. We may construct the vector pair $(\mathbf{q}, \mathbf{w})$ from $\mathbf{v}$, where $\mathbf{q}$ is a vector of states of the input transducer and $\mathbf{w}$ is a vector of residual weights, by choosing a (state, weight) pair from each output state in $\mathbf{v}$. For example, let $Q_1 = \{a, b, c\}$. Then two possible output transducer states could be $x = ((a, 1))$ and $y = ((a, 0.25), (c, 0.75))$. If we choose $\mathbf{v} = \langle y, x \rangle$ then a valid vector pair $(\mathbf{q}, \mathbf{w})$ is $\mathbf{q} = \langle c, a \rangle$, $\mathbf{w} = \langle 0.75, 1 \rangle$.

The sets $\Gamma(\mathbf{v}, x)$, $\gamma(\mathbf{v}, x)$, and $\nu(\mathbf{v}, x)$ are defined as follows:

- $\Gamma(\mathbf{v}, x) = \{(\mathbf{q}, \mathbf{w}) \text{ from } \mathbf{v} : \exists e = (\mathbf{q}, x, \delta_1(e), \sigma_1(e)) \in E_1\}$.
- $\gamma(\mathbf{v}, x) = \{(\mathbf{q}, \mathbf{w}, e) \text{ from } \mathbf{v} \times E_1 : \exists e = (\mathbf{q}, x, \delta_1(e), \sigma_1(e)) \in E_1\}$.
- $\nu(\mathbf{v}, x) = \{q' : \exists (\mathbf{q}, \mathbf{w}) \text{ from } \mathbf{v}, \exists e = (\mathbf{q}, x, q', \sigma_1(e)) \in E_1\}$.

.

$\Gamma(\mathbf{v}, x)$ is the set of vector pairs $(\mathbf{q}, \mathbf{w})$ constructed from $\mathbf{v}$ where each $\mathbf{q}$ is an input vector in a transition $e \in E_1$ with label x. $\gamma(\mathbf{v}, x)$ is the set of unique transitions e paired with the appropriate pair for each $(\mathbf{q}, \mathbf{w})$ in $\Gamma(\mathbf{v}, x)$. $\nu(\mathbf{v}, x)$ is the set of states reachable from the transitions in $\Gamma(\mathbf{v}, x)$.

The consideration of *vectors* of states on the incident edge of transitions effects two noticeable changes on the algorithm as it is presented in (Mohri, 1997). The first, relatively trivial, change is the inclusion of the residual of multiple states in the calculation of weights and residuals on lines 16 and 17. The second change is the production of vectors for consideration. Whereas the string-based algorithm considered newly-created states in turn, we must consider newly-available vectors. For each newly created state, newly available vectors can be formed by using that state with the other states of the output transducer. This operation is performed on lines 7 and 22 of the algorithm.

5 Empirical Studies

We now turn to some empirical studies. We examine the practical impact of the presented work by showing:

- That the multiple derivation problem is pervasive in practice and determinization is effective at removing duplicate trees.
- That duplication causes misleading weighting of individual trees and the summing achieved from weighted determinization corrects this error, leading to re-ordering of the n-best list.
- That weighted determinization positively affects end-to-end system performance.

We also compare our results to a commonly used technique for estimation of n-best lists, i.e., summing over the top $k > n$ derivations to get weight estimates of the top $m \leq k$ unique elements.

5.1 Machine translation

We obtain packed-forest English outputs from 116 short Chinese sentences computed by a string-to-tree machine translation system based on (Galley, et. al., 2004). The system is trained on all Chinese-English parallel data available from the Linguistic Data Consortium. The decoder for this system is a CKY algorithm that negotiates the space described in (DeNeefe, et. al., 2005). No language model was used in this experiment.

The forests contain a median of 1.4×10^{12} English parse trees each. We remove cycles from each

Algorithm 1: Weighted Determinization of Tree Automata

Input: **BOTTOM-UP TREE-TO-WEIGHT TRANSDUCER** $\tau_1 = (Q_1, \Sigma, i_1, F_1, E_1, \lambda_1, \rho_1)$.
Output: **SUBSEQUENTIAL BOTTOM-UP TREE-TO-WEIGHT TRANSDUCER** $\tau_2 = (Q_2, \Sigma, i_2, F_2, E_2, \lambda_2, \rho_2)$.

begin
  $\lambda_2 \longleftarrow \lambda_1(i_1)$
  $i_2 \longleftarrow \{(i_1, 1)\}$
  PRIORITY QUEUE $P \longleftarrow \emptyset$
  $Q_2 \longleftarrow \{i_2\}$
  $F_2 \longleftarrow \emptyset$
  ENQUEUE$(P, \langle i_2 \rangle)$
  while $P \neq \emptyset$ **do**
    $\mathbf{v} \longleftarrow$ *head*$[P]$
    $k \longleftarrow |\mathbf{v}|$
    for *each* $p \in \boldsymbol{v}$ *such that* $p \notin F_2$ **do**
      if $\exists (q, w) \in p$ *such that* $q \in F_1$ **then**
        $\rho_2(p) \longleftarrow \sum_{(q,w) \in p \text{ s.t. } q \in F_1} w \times \rho_1(q)$
        $F_2 \longleftarrow F_2 \cup p$
    for *each* $x \in \Sigma^{(k)}$ *such that* $\Gamma(\boldsymbol{v}, x) \neq \emptyset$ **do**
      $\sigma_2(\mathbf{v}, x) \longleftarrow \sum_{((q_1,w_1),\ldots,(q_i,w_i)) \in \Gamma(\mathbf{v},x)} [w_1 \times \ldots \times w_i \times \sum_{e=(\langle q_1,\ldots,q_i \rangle, x, \delta_1(e), \sigma_1(e)) \in E_1} \sigma_1(e)]$
      $\delta_2(\mathbf{v}, x) \longleftarrow \bigcup_{q' \in \nu(\mathbf{v},x)} \{(q', \frac{1}{\sigma_2(\mathbf{v},x)} \times \sum_{((q_1,w_1),\ldots,(q_i,w_i),e) \in \gamma(\mathbf{v},x), \text{ s.t. } \delta_1(e) = q'} w_1 \times \ldots \times w_i \times \sigma_1(e))\}$
      $E_2 \longleftarrow E_2 \cup (\mathbf{v}, x, \delta_2(\mathbf{v}, x), \sigma_2(\mathbf{v}, x))$
      `/* RANK(`i`) returns the largest hyperedge size that can leave state` i`.`
      `COMBINATIONS(`Q, N, k`) returns all possible vectors of length 1..`k
      `containing members of` Q `and at least one member of` N`. */`
      if $\delta_2(\boldsymbol{v}, x)$ *is a new state* **then**
        for *each* $\boldsymbol{u} \in$ **COMBINATIONS**$(Q_2, \{\delta_2(\boldsymbol{v}, x)\}, \max_{q \in p \in Q_2 \cup \{\delta_2(\boldsymbol{v},x)\}}$ **RANK**$(q))$ **do**
          if $\boldsymbol{u}$ *is a new vector* **then**
            ENQUEUE$(P, \mathbf{u})$
        $Q_2 \longleftarrow Q_2 \cup \{\delta_2(\mathbf{v}, x)\}$
    DEQUEUE(P)
end

forest,[3] apply our determinization algorithm, and extract the n-best trees using a variant of (Huang and Chiang, 2005). The effects of weighted determinization on an n-best list are obvious to casual inspection. Figure 7 shows the improvement in quality of the top 10 trees from our example translation after the application of the determinization algorithm.

The improvement observed circumstantially holds up to quantitative analysis as well. The forests obtained by the determinized grammars have between 1.39% and 50% of the number of trees of their undeterminized counterparts. On average, the determinized forests contain 13.7% of the original number of trees. Since a determinized forest contains no repeated trees but contains exactly the same unique trees as its undeterminized counterpart, this indicates that an average of 86.3% of the trees in an undeterminized MT output forest are duplicates.

Weighted determinization also causes a surprisingly large amount of n-best reordering. In 77.6% of the translations, the tree regarded as "best" is different after determinization. This means that in a large majority of cases, the tree with the highest weight is not recognized as such in the undeterminized list because its weight is divided among its multiple derivations. Determinization allows these instances and their associated weights to combine and puts the highest weighted tree, not the highest weighted derivation, at the top of the list.

[3] As in (Mohri, 1997), determinization may be applicable to some automata that recognize infinite languages. In practice, cycles in tree automata of MT results are almost never desired, since these represent recursive insertion of words.

method	Bleu
undeterminized	21.87
top-500 "crunching"	23.33
determinized	24.17

Figure 6: Bleu results from string-to-tree machine translation of 116 short Chinese sentences with no language model. The use of best derivation (undeterminized), estimate of best tree (top-500), and true best tree (determinized) for selection of translation is shown.

We can compare our method with the more commonly used methods of "crunching" k-best lists, where $k > n$. The duplicate sentences in the k trees are combined, hopefully resulting in at least n unique members with an estimation of the true tree weight for each unique tree. Our results indicate this is a rather crude estimation. When the top 500 derivations of the translations of our test corpus are summed, only 50.6% of them yield an estimated highest-weighted tree that is the same as the true highest-weighted tree.

As a measure of the effect weighted determinization and its consequential re-ordering has on an actual end-to-end evaluation, we obtain Bleu scores for our 1-best translations from determinization, and compare them with the 1-best translations from the undeterminized forest and the 1-best translations from the top-500 "crunching" method. The results are tabulated in Figure 6. Note that in 26.7% of cases determinization did not terminate in a reasonable amount of time. For these sentences we used the best parse from top-500 estimation instead. It is not surprising that determinization may occasionally take a long time; even for a language of monadic trees (i.e. strings) the determinization algorithm is NP-complete, as implied by (Casacuberta and de la Higuera, 2000) and, e.g. (Dijkstra, 1959).

5.2 Data-Oriented Parsing

Weighted determinization of tree automata is also useful for parsing. Data-Oriented Parsing (DOP)'s methodology is to calculate weighted derivations, but as noted in (Bod, 2003), it is the highest ranking parse, not derivation, that is desired. Since (Sima'an, 1996) showed that finding the highest ranking parse is an NP-complete problem, it has been common to estimate the highest ranking parse by the previously described "crunching" method.

method	Recall	Precision	F-measure
undeterminized	80.23	80.18	80.20
top-500 "crunching"	80.48	80.29	80.39
determinized	81.09	79.72	80.40

Figure 8: Recall, precision, and F-measure results on DOP-style parsing of section 23 of the Penn Treebank. The use of best derivation (undeterminized), estimate of best tree (top-500), and true best tree (determinized) for selection of parse output is shown.

We create a DOP-like parsing model[4] by extracting and weighting a subset of subtrees from sections 2-21 of the Penn Treebank and use a DOP-style parser to generate packed forest representations of parses of the 2416 sentences of section 23. The forests contain a median of 2.5×10^{15} parse trees. We then remove cycles and apply weighted determinization to the forests. The number of trees in each determinized parse forest is reduced by a factor of between 2.1 and 1.7×10^{14}. On average, the number of trees is reduced by a factor of 900,000, demonstrating a much larger number of duplicate parses prior to determinization than in the machine translation experiment. The top-scoring parse after determinization is different from the top-scoring parse before determinization for 49.1% of the forests, and when the determinization method is "approximated" by crunching the top-500 parses from the undeterminized list only 55.9% of the top-scoring parses are the same, indicating the crunching method is not a very good approximation of determinization. We use the standard F-measure combination of recall and precision to score the top-scoring parse in each method against reference parses. The results are tabulated in Figure 8. Note that in 16.9% of cases determinization did not terminate. For those sentences we used the best parse from top-500 estimation instead.

6 Conclusion

We have shown that weighted determinization is useful for recovering n-best unique trees from a weighted forest. As summarized in Figure 9, the

[4]This parser acquires a small subset of subtrees, in contrast with DOP, and the beam search for this problem has not been optimized.

```
31.87:  S(NP-C(NPB(DT(this) NNS(cases))) VP(VBD(had) VP-C(VBN(aroused) NP-C(NPB(DT(the) JJ(american) NNS(protests))))) .(.))
32.11:  S(NP-C(NPB(DT(this) NNS(cases))) VP(VBD(had) VP-C(VBN(caused) NP-C(NPB(DT(the) JJ(american) NNS(protests))))) .(.))
32.15:  S(NP-C(NPB(DT(this) NNS(cases))) VP(VBD(had) VP-C(VB(arouse) NP-C(NPB(DT(the) JJ(american) NNS(protests))))) .(.))
32.55:  S(NP-C(NPB(DT(this) NNS(cases))) VP(VBD(had) VP-C(VB(cause) NP-C(NPB(DT(the) JJ(american) NNS(protests))))) .(.))
32.60:  S(NP-C(NPB(DT(this) NNS(cases))) VP(VBD(had) VP-C(VBN(attracted) NP-C(NPB(DT(the) JJ(american) NNS(protests))))) .(.))
33.16:  S(NP-C(NPB(DT(this) NNS(cases))) VP(VBD(had) VP-C(VB(provoke) NP-C(NPB(DT(the) JJ(american) NNS(protests))))) .(.))
33.27:  S(NP-C(NPB(DT(this) NNS(cases))) VP(VBG(causing) NP-C(NPB(DT(the) JJ(american) NNS(protests)))) .(.))
33.29:  S(NP-C(NPB(DT(this) NN(case))) VP(VBD(had) VP-C(VBN(aroused) NP-C(NPB(DT(the) JJ(american) NNS(protests))))) .(.))
33.31:  S(NP-C(NPB(DT(this) NNS(cases))) VP(VBD(had) VP-C(VBN(aroused) NP-C(NPB(DT(the) NN(protest)) PP(IN(of) NP-C(NPB(DT(the)
NNS(united_states)))))) .(.))
33.33:  S(NP-C(NPB(DT(this) NNS(cases))) VP(VBD(had) VP-C(VBN(incurred) NP-C(NPB(DT(the) JJ(american) NNS(protests))))) .(.))
```

Figure 7: Ranked list of machine translation results with no repeated trees.

experiment	undeterminized	determinized
machine translation	1.4×10^{12}	2.0×10^{11}
parsing	2.5×10^{15}	2.3×10^{10}

Figure 9: Median trees per sentence forest in machine translation and parsing experiments before and after determinization is applied to the forests, removing duplicate trees.

number of repeated trees prior to determinization was typically very large, and thus determinization is critical to recovering true tree weight. We have improved evaluation scores by incorporating the presented algorithm into our MT work and we believe that other NLP researchers working with trees can similarly benefit from this algorithm.

Further advances in determinization will provide additional benefit to the community. The translation system detailed here is a string-to-tree system, and the determinization algorithm returns the n-best unique trees from a packed forest. Users of MT systems are generally interested in the string yield of those trees, and not the trees per se. Thus, an algorithm that can return the n-best unique *strings* from a packed forest would be a useful extension.

We plan for our weighted determinization algorithm to be one component in a generally available tree automata package for intersection, composition, training, recognition, and generation of weighted and unweighted tree automata for research tasks such as the ones described above.

Acknowledgments

We thank Liang Huang for fruitful discussions which aided in this work and David Chiang, Daniel Marcu, and Steve DeNeefe for reading an early draft and providing useful comments. This work was supported by NSF grant IIS-0428020.

References

Rens Bod. 1992. A Computational model of language performance: data oriented parsing. In *Proc. COLING*

Rens Bod. 2003. An efficient implementation of a new DOP model. In *Proc. EACL*,

Björn Borchardt and Heiko Vogler. 2003. Determinization of finite state weighted tree automata. *Journal of Automata, Languages and Combinatorics*, 8(3).

W. S. Brainerd. 1969. Tree generating regular systems. *Information and Control*, 14.

F. Casacuberta and C. de la Higuera. 2000. Computational complexity of problems on probabilistic grammars and transducers. In *Proc. ICGI*.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proc. ACL*.

H. Comon and M. Dauchet and R. Gilleron and F. Jacquemard and D. Lugiez and S. Tison and M. Tommasi. 1997 *Tree Automata Techniques and Applications*.

S. DeNeefe and K. Knight and H. Chan. 2005. Interactively exploring a machine translation model. Poster in *Proc. ACL*.

Edsger W. Dijkstra 1959. A note on two problems in connexion with graphs *Numerische Mathematik*, 1.

J. E. Doner 1970. Tree acceptors and some of their applications *J. Comput. System Sci.*, 4.

M. Galley and M. Hopkins and K. Knight and D. Marcu. 2004. What's in a translation rule? In *Proc. HLT-NAACL*.

Ferenc Gécseg and Magnus Steinby 1984. *Tree Automata*. Akadémiai Kiadó, Budapest.

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

Irene Langkilde and Kevin Knight 1998 The Practical Value of N-Grams in Generation In *Proc. INLG*.

M. Magidor and G. Moran. 1969. Finite automata over finite trees *Technical Report 30*. Hebrew University, Jerusalem.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2).

Mehryar Mohri and Michael Riley. 2002. An efficient algorithm for the n-best strings problem. In *Proc. ICSLP*.

M. O. Rabin. 1969. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141.

Yves Schabes. 1990. *Mathematical and computational aspects of lexicalized grammars*. Ph.D. thesis. University of Pennsylvania, Philadelphia, PA.

Khalil Sima'an. 1996. Computational complexity of probabilistic disambiguation by means of tree-grammars. In *Proc. COLING*.

J. W. Thatcher and J. B. Wright. 1968. Generalized finite automata theory with an application to a decision problem of second order logic. *Mathematical Systems Theory*, 2.

Aggregation via Set Partitioning for Natural Language Generation

Regina Barzilay
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
regina@csail.mit.edu

Mirella Lapata
School of Informatics
University of Edinburgh
mlap@inf.ed.ac.uk

Abstract

The role of aggregation in natural language generation is to combine two or more linguistic structures into a single sentence. The task is crucial for generating concise and readable texts. We present an efficient algorithm for automatically learning aggregation rules from a text and its related database. The algorithm treats aggregation as a set partitioning problem and uses a global inference procedure to find an optimal solution. Our experiments show that this approach yields substantial improvements over a clustering-based model which relies exclusively on local information.

1 Introduction

Aggregation is an essential component of many natural language generation systems (Reiter and Dale, 2000). The task captures a mechanism for merging together two or more linguistic structures into a single sentence. Aggregated texts tend to be more concise, coherent, and more readable overall (Dalianis, 1999; Cheng and Mellish, 2000). Compare, for example, sentence (2) in Table 1 and its non-aggregated counterpart in sentences (1a)–(1d). The difference between the fluent aggregated sentence and its abrupt and redundant alternative is striking.

The benefits of aggregation go beyond making texts less stilted and repetitive. Researchers in psycholinguistics have shown that by eliminating redundancy, aggregation facilitates text comprehension and recall (see Yeung (1999) and the references therein). Furthermore, Di Eugenio et al. (2005) demonstrate that aggregation can improve learning in the context of an intelligent tutoring application.

(1)	a.	Holocomb had an incompletion in the first quarter.
	b.	Holocomb had another incompletion in the first quarter.
	c.	Davis was among four San Francisco defenders.
	d.	Holocomb threw to Davis for a leaping catch.
(2)	**After two incompletions in the first quarter, Holcomb found Davis among four San Francisco defenders for a leaping catch.**	

Table 1: Aggregation example (in boldface) from a corpus of football summaries

In existing generation systems, aggregation typically comprises two processes: semantic grouping and sentence structuring (Wilkinson, 1995). The first process involves partitioning semantic content (usually the output of a content selection component) into disjoint sets, each corresponding to a single sentence. The second process is concerned with syntactic or lexical decisions that affect the realization of an aggregated sentence.

To date, this task has involved human analysis of a domain-relevant corpus and manual development of aggregation rules (Dalianis, 1999; Shaw, 1998). The corpus analysis and knowledge engineering work in such an approach is substantial, prohibitively so in

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 359–366,
New York, June 2006.

large domains. But since corpus data is already used in building aggregation components, an appealing alternative is to try and learn the rules of semantic grouping directly from the data. Clearly, this would greatly reduce the human effort involved and ease porting generation systems to new domains.

In this paper, we present an automatic method for performing the semantic grouping task. We address the following problem: given an aligned parallel corpus of sentences and their underlying semantic representations, how can we learn grouping constraints automatically? In our case the semantic content corresponds to entries from a database; however, our algorithm could be also applied to other representations such as propositions or sentence plans.

We formalize semantic grouping as a set partitioning problem, where each partition corresponds to a sentence. The strength of our approach lies in its ability to capture global partitioning constraints by performing collective inference over local pairwise assignments. This design allows us to integrate important constraints developed in symbolic approaches into an automatic aggregation framework. At a *local* level, pairwise constraints capture the semantic compatibility between pairs of database entries. For example, if two entries share multiple attributes, then they are likely to be aggregated. Local constraints are learned using a binary classifier that considers all pairwise combinations attested in our corpus. At a *global* level, we search for a semantic grouping that maximally agrees with the pairwise preferences while simultaneously satisfying constraints on the partitioning as a whole. Global constraints, for instance, could prevent the creation of overly long sentences, and, in general, control the compression rate achieved during aggregation. We encode the global inference task as an integer linear program (ILP) that can be solved using standard optimization tools.

We evaluate our approach in a sports domain represented by large real-world databases containing a wealth of interrelated facts. Our aggregation algorithm model achieves an 11% F-score increase on grouping entry pairs over a greedy clustering-based model which does not utilize global information for the partitioning task. Furthermore, these results demonstrate that aggregation is amenable to an automatic treatment that does not require human involvement.

In the following section, we provide an overview of existing work on aggregation. Then, we define the learning task and introduce our approach to content grouping. Next, we present our experimental framework and data. We conclude the paper by presenting and discussing our results.

2 Related Work

Due to its importance in producing coherent and fluent text, aggregation has been extensively studied in the text generation community.[1] Typically, semantic grouping and sentence structuring are interleaved in one step, thus enabling the aggregation component to operate over a rich feature space. The common assumption is that other parts of the generation system are already in place during aggregation, and thus the aggregation component has access to discourse, syntactic, and lexical constraints.

The interplay of different constraints is usually captured by a set of hand-crafted rules that guide the aggregation process (Scott and de Souza, 1990; Hovy, 1990; Dalianis, 1999; Shaw, 1998). Alternatively, these rules can be learned from a corpus. For instance, Walker et al. (2001) propose an overgenerate-and-rank approach to aggregation within the context of a spoken dialog application. Their system relies on a preference function for selecting an appropriate aggregation among multiple alternatives and assumes access to a large feature space expressing syntactic and pragmatic features of the input representations. The preference function is learned from a corpus of candidate aggregations marked with human ratings. Another approach is put forward by Cheng and Mellish (2000) who use a genetic algorithm in combination with a hand-crafted preference function to opportunistically find a text that satisfies aggregation and planning constraints.

Our approach differs from previous work in two important respects. First, our ultimate goal is a generation system which can be entirely induced from a parallel corpus of sentences and their corresponding database entries. This means that our generator will operate over more impoverished representations than are traditionally assumed. For example we do

[1] The approaches are too numerous to list; we refer the interested reader to Reiter and Dale (2000) and Reape and Mellish (1999) for comprehensive overviews.

Passing					
PLAYER	CP/AT	**YDS**	**AVG**	**TD**	INT
Cundiff	22/37	237	6.4	1	1
Carter	23/47	237	5.0	1	4
...	...	...	...	...	...

Rushing					
PLAYER	REC	**YDS**	**AVG**	LG	**TD**
Hambrick	13	33	2.5	10	1
...	...	...	...	...	...

1	(*Passing* (Cundiff 22/37 237 6.4 1 1))
	(*Passing* (Carter 23/47 237 5.0 1 4))
2	(*Interception* (Lindell 1 52 1))
	(*Kicking* (Lindell 3/3 100 38 1/1 10))
3	(*Passing* (Bledsoe 17/34 104 3.1 0 0))
4	(*Passing* (Carter 15/32 116 3.6 1 0))
5	(*Rushing* (Hambrick 13 33 2.5 10 1))
6	(*Fumbles* (Bledsoe 2 2 0 0 0))

Table 2: Excerpt of database and (simplified) example of aggregated entries taken from a football domain. This fragment will give rise to 6 sentences in the final text.

not presume to know all possible ways in which our database entries can be lexicalized, nor do we presume to know which semantic or discourse relations exist between different entries. In this framework, aggregation is the task of grouping semantic content without making any decisions about sentence structure or its surface realization. Second, we strive for an approach to the aggregation problem which is as domain- and representation-independent as possible.

3 Problem Formulation

We formulate aggregation as a *supervised partitioning task*, where the goal is to find a clustering of input items that maximizes a global utility function. The input to the model consists of a set E of database entries selected by a content planner. The output of the model is a partition $S = \{S_i\}$ of nonempty subsets such that each element of E appears in exactly one subset.[2] In the context of aggregation, each partition represents entries that should be verbalized in the same sentence. An example of a partitioning is illustrated in the right side of Table 2 where eight entries are partitioned into six clusters.

We assume access to a relational database where each entry has a type and a set of attributes associated with it. Table 2 (left) shows an excerpt of the database we used for our experiments. The aggregated text in Table 2 (right) contains entries of five types: `Passing`, `Interception`, `Kicking`, `Rushing`, and `Fumbles`. Entries of type `Passing` have six attributes — `PLAYER`, `CP/AT`, `YDS`, `AVG`, `TD`, `INT`, entries of type `Interception` have four attributes, and so on. We assume the existence of a non-empty set of attributes that we can use for meaningful comparison between entities of different types. In the example above, types `Passing` and `Rushing` share the attributes `PLAYER`, `AVG` (short for average), `TD` (short for touchdown) and `YDS` (short for yards). These are indicated in boldface in Table 2. In Section 4.1, we discuss how a set of shared attributes can be determined for a given database.

Our training data consists of entry sets with a known partitioning. During testing, our task is to infer a partitioning for an unseen set of entries.

4 Modeling

Our model is inspired by research on text aggregation in the natural language generation community (Cheng and Mellish, 2000; Shaw, 1998). A common theme across different approaches is the notion of similarity — content elements described in the same sentence should be related to each other in some meaningful way to achieve conciseness and coherence. Consider for instance the first cluster in Table 2. Here, we have two entries of the same type (i.e., `Passing`). Furthermore, the entries share the same values for the attributes `YDS` and `TD` (i.e., 237 and 1). On the other hand, clusters 5 and 6 have no attributes in common. This observation motivates modeling aggregation as a binary classification task: given a pair of entries, predict their aggregation status based on the similarity of their attributes. Assuming a perfect classifier, pairwise assignments

[2] By definition, a partitioning of a set defines an equivalence relation which is reflexive, symmetric, and transitive.

will be consistent with each other and will therefore yield a valid partitioning.

In reality, however, this approach may produce globally inconsistent decisions since it treats each pair of entries in isolation. Moreover, a pairwise classification model cannot express general constraints regarding the partitioning as a whole. For example, we may want to constrain the size of the generated partitions and the compression rate of the document, or the complexity of the generated sentences.

To address these requirements, our approach relies on global inference. Given the pairwise predictions of a *local* classifier, our model finds a *globally optimal* assignment that satisfies partitioning-level constraints. The computational challenge lies in the complexity of such a model: we need to find an optimal partition in an exponentially large search space. Our approach is based on an Integer Linear Programming (ILP) formulation which can be effectively solved using standard optimization tools. ILP models have been successfully applied in several natural language processing tasks, including relation extraction (Roth and Yih, 2004), semantic role labeling (Punyakanok et al., 2004) and the generation of route directions (Marciniak and Strube, 2005).

In the following section, we introduce our local pairwise model and afterward we present our global model for partitioning.

4.1 Learning Pairwise Similarity

Our goal is to determine whether two database entries should be aggregated given the similarity of their shared attributes. We generate the training data by considering all pairs $\langle e_i, e_j \rangle \in E \times E$, where E is the set of all entries attested in a given document. An entry pair forms a positive instance if its members belong to the same partition in the training data. For example, we will generate $\frac{8\times7}{2}$ unordered entry pairs for the eight entries from the document in Table 2. From these, only two pairs constitute positive instances, i.e., clusters 1 and 2. All other pairs form negative instances.

The computation of pairwise similarity is based on the attribute set $\mathcal{A} = \{A_i\}$ shared between the two entries in the pair. As discussed in Section 3, the same attributes can characterize multiple entry types, and thus form a valid basis for entry comparison. The shared attribute set $\mathcal{A}$ could be identified in many ways. For example, using domain knowledge or by selecting attributes that appear across multiple types. In our experiments, we follow the second approach: we order attributes by the number of entry types in which they appear, and select the top five[3].

A pair of entries is represented by a binary feature vector $\{x_i\}$ in which coordinate x_i indicates whether two entries have the same value for attribute i. The feature vector is further expanded by conjuctive features that explicitly represent overlap in values of multiple attributes up to size k. The parameter k controls the cardinality of the maximal conjunctive set and is optimized on the development set.

To illustrate our feature generation process, consider the pair (*Passing* (Quincy Carter 15/32 116 3.6 1 0)) and (*Rushing* (Troy Hambrick 13 33 2.5 10 1)) from Table 2. Assuming $\mathcal{A} = \{\texttt{Player}, \texttt{Yds}, \texttt{TD}\}$ and $k = 2$, the similarity between the two entries will be expressed by six features, three representing overlap in individual attributes and three representing overlap when considering pairs of attributes. The resulting feature vector has the form $\langle 0, 0, 1, 0, 0, 0 \rangle$.

Once we define a mapping from database entries to features, we employ a machine learning algorithm to induce a classifier on the feature vectors generated from the training documents. In our experiments, we used a publicly available maximum entropy classifier[4] for this task.

4.2 Partitioning with ILP

Given the pairwise predictions of the local classifier, we wish to find a valid global partitioning for the entries in a single document. We thus model the interaction between *all* pairwise aggregation decisions as an optimization problem.

Let $c_{\langle e_i, e_j \rangle}$ be the probability of seeing entry pair $\langle e_i, e_j \rangle$ aggregated (as computed by the pairwise classifier). Our goal is to find an assignment that maximizes the sum of pairwise scores and forms a valid partitioning. We represent an assignment using a set of indicator variables $x_{\langle e_i, e_j \rangle}$ that are set

[3] Selecting a larger number of attributes for representing similarity would result in considerably sparser feature vectors.

[4] The software can be downloaded from `http://www.isi.edu/~hdaume/megam/index.html`.

to 1 if $\langle e_i, e_j \rangle$ is aggregated, and 0 otherwise. The score of a global assignment is the sum of its pairwise scores:

$$\sum_{\langle e_i,e_j\rangle \in E \times E} c_{\langle e_i,e_j\rangle} x_{\langle e_i,e_j\rangle} + (1 - c_{\langle e_i,e_j\rangle})(1 - x_{\langle e_i,e_j\rangle}) \quad (1)$$

Our inference task is solved by maximizing the overall score of pairs in a given document:

$$\text{argmax} \sum_{\langle e_i,e_j\rangle \in E \times E} c_{\langle e_i,e_j\rangle} x_{\langle e_i,e_j\rangle} + (1 - c_{\langle e_i,e_j\rangle})(1 - x_{\langle e_i,e_j\rangle}) \quad (2)$$

subject to:

$$x_{\langle e_i,e_j\rangle} \in \{0, 1\} \; \forall \, e_i, e_j \in E \times E \quad (3)$$

We augment this basic formulation with two types of constraints. The first type of constraint ensures that pairwise assignments lead to a consistent partitioning, while the second type expresses global constraints on partitioning.

Transitivity Constraints We place constraints that enforce transitivity in the label assignment: if $x_{\langle e_i,e_j\rangle} = 1$ and $x_{\langle e_j,e_k\rangle} = 1$, then $x_{\langle e_i,e_k\rangle} = 1$. A pairwise assignment that satisfies this constraint defines an equivalence relation, and thus yields a unique partitioning of input entries (Cormen et al., 1992).

We implement transitivity constraints by introducing for every triple e_i, e_j, e_k $(i \neq j \neq k)$ an inequality of the following form:

$$x_{\langle e_i,e_k\rangle} \geq x_{\langle e_i,e_j\rangle} + x_{\langle e_j,e_k\rangle} - 1 \quad (4)$$

If both $x_{\langle e_i,e_j\rangle}$ and $x_{\langle e_j,e_k\rangle}$ are set to one, then $x_{\langle e_i,e_k\rangle}$ also has to be one. Otherwise, $x_{\langle e_i,e_k\rangle}$ can be either 1 or 0.

Global Constraints We also want to consider global document properties that influence aggregation. For example, documents with many database entries are likely to exhibit different compression rates during aggregation when compared to documents that contain only a few.

Our first global constraint controls the number of aggregated sentences in the document. This is achieved by limiting the number of entry pairs with positive labels for each document:

$$\sum_{\langle e_i,e_j\rangle \in E \times E} x_{\langle e_i,e_j\rangle} \leq m \quad (5)$$

Notice that the number m is not known in advance. However, we can estimate this parameter from our development data by considering documents of similar size (as measured by the number of corresponding entry pairs.) For example, texts with thousand entry pairs contain on average 70 positive labels, while documents with 200 pairs have around 20 positive labels. Therefore, we set m separately for every document by taking the average number of positive labels observed in the development data for the document size in question.

The second set of constraints controls the length of the generated sentences. We expect that there is an upper limit on the number of pairs that can be clustered together. This restriction can be expressed in the following form:

$$\forall \, e_i \sum_{e_j \in E} x_{\langle e_i,e_j\rangle} \leq k \quad (6)$$

This constraint ensures that there can be at most k positively labeled pairs for any entry e_i. In our corpus, for instance, at most nine entries can be aggregated in a sentence. Again k is estimated from the development data by taking into account the average number of positively labeled pairs for every entry type (see Table 2). We therefore indirectly capture the fact that some entry types (e.g., `Passing`) are more likely to be aggregated than others (e.g., `Kicking`).

Solving the ILP In general, solving an integer linear program is NP-hard (Cormen et al., 1992). Fortunately, there exist several strategies for solving ILPs. In our study, we employed *lp_solve*, an efficient Mixed Integer Programming solver[5] which implements the Branch-and-Bound algorithm. We generate and solve an ILP for every document we wish to aggregate. Documents of average size (approximately 350 entry pairs) take under 30 minutes on a 450 MHz Pentium III machine.

[5]The software is available from `http://www.geocities.com/lpsolve/`

5 Evaluation Set-up

The model presented in the previous section was evaluated in the context of generating summary reports for American football games. In this section we describe the corpus used in our experiments, our procedure for estimating the parameters of our models, and the baseline method used for comparison with our approach.

Data For training and testing our algorithm, we employed a corpus of football game summaries collected by Barzilay and Lapata (2005). The corpus contains 468 game summaries from the official site of the American National Football League[6] (NFL). Each summary has an associated database containing statistics about individual players and events. In total, the corpus contains 73,400 database entries, 7.1% of which are verbalized; each entry is characterized by a type and a set of attributes (see Table 2). Database entries are automatically aligned with their corresponding sentences in the game summaries by a procedure that considers anchor overlap between entity attributes and sentence tokens. Although the alignment procedure is relatively accurate, there is unavoidably some noise in the data.

The distribution of database entries per sentence is shown in Figure 1. As can be seen, most aggregated sentences correspond to two or three database entries. Each game summary contained 14.3 entries and 9.1 sentences on average. The training and test data were generated as described in Section 4.1. We used 96,434 instances (300 summaries) for training, 59,082 instances (68 summaries) for testing, and 53,776 instances (100 summaries) for development purposes.

Parameter Estimation As explained in Section 4, we infer a partitioning over a set of database entries in a two-stage process. We first determine how likely all entry pairs are to be aggregated using a local classifier, and then infer a valid global partitioning for all entries. The set of shared attributes $\mathcal{A}$ consists of five features that capture overlap in players, time (measured by game quarters), action type, outcome type, and number of yards. The maximum cardinality of the set of conjunctive features is five.

[6] See `http://www.nfl.com/scores`.

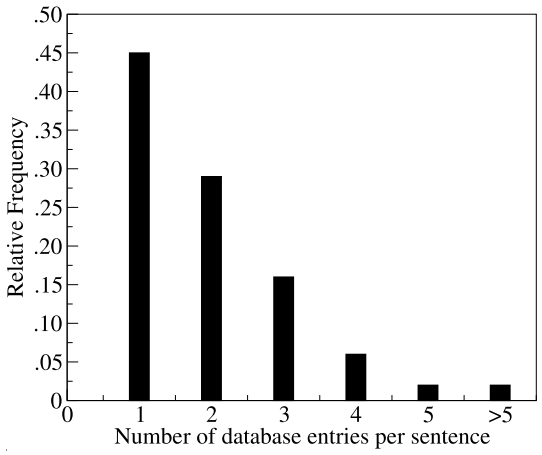


Figure 1: Distribution of aggregated sentences in the NFL corpus

Overall, our local classifier used 28 features, including 23 conjunctive ones. The maximum entropy classifier was trained for 100 iterations. The global constraints for our ILP models are parametrized (see equations (5) and (6)) by m and k which are estimated separately for every test document. The values of m ranged from 2 to 130 and for k from 2 to 9.

Baseline Clustering is a natural baseline model for our partitioning problem. In our experiments, we a employ a single-link agglomerative clustering algorithm that uses the scores returned by the maximum entropy classifier as a pairwise distance measure. Initially, the algorithm creates a separate cluster for each sentence. During each iteration, the two closest clusters are merged. Again, we do not know in advance the appropriate number of clusters for a given document. This number is estimated from the training data by averaging the number of sentences in documents of the same size.

Evaluation Measures We evaluate the performance of the ILP and clustering models by measuring F-score over pairwise label assignments. We compute F-score individually for each document and report the average. In addition, we compute partition accuracy in order to determine how many sentence-level aggregations our model predicts correctly.

Clustering	Precision	Recall	F-score
Mean	57.7	66.9	58.4
Min	0.0	0.0	0.0
Max	100.0	100.0	100.0
StDev	28.2	23.9	23.1

ILP Model	Precision	Recall	F-score
Mean	82.2	65.4	70.3
Min	37.5	28.6	40.0
Max	100.0	100.0	100.0
StDev	19.2	20.3	16.6

Table 3: Results on pairwise label assignment (precision, recall, and F-score are averaged over documents); comparison between clustering and ILP models

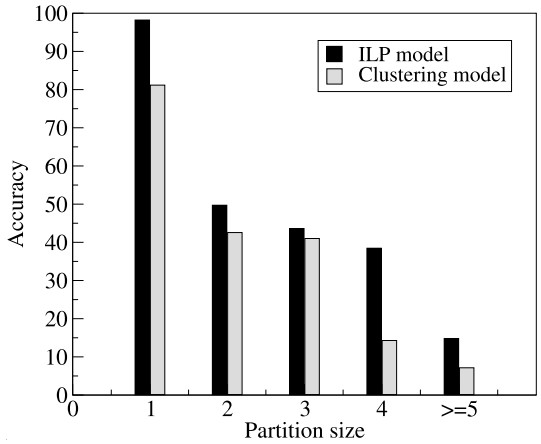


Figure 2: Partition accuracy for sentences of different size

6 Results

Our results are summarized in Table 3. As can be seen, the ILP model outperforms the clustering model by a wide margin (11.9% F-score). The two methods yield comparable recall; however, the clustering model lags considerably behind as far as precision is concerned (the difference is 24.5 %).[7]

Precision is more important than recall in the context of our aggregation application. Incorrect aggregations may have detrimental effects on the coherence of the generated text. Choosing not to aggregate may result in somewhat repetitive texts; however, the semantic content of the underlying text remains intact. In the case of wrong aggregations, we may group together facts that are not compatible, and even introduce implications that are false.

We also consider how well our model performs when evaluated on total partition accuracy. Here, we are examining the partitioning as a whole and ask the following question: how many clusters of size 1, 2 ... n did the algorithm get right? This evaluation measure is stricter than F-score which is computed over pairwise label assignments. The partition accuracy for entry groups of varying size is shown in Figure 2. As can be seen, in all cases the ILP outperforms the clustering baseline. Both models are fairly accurate at identifying singletons, i.e., database entries which are not aggregated. Performance is naturally worse when considering larger clusters. Interestingly, the difference between the two models becomes more pronounced for partition sizes 4 and 5 (see Figure 2). The ILP's accuracy increases by 24% for size 4 and 8% for size 5.

These results empirically validate the importance of global inference for the partitioning task. Our formulation allows us to incorporate important document-level constraints as well as consistency constraints which cannot be easily represented in a vanilla clustering model.

7 Conclusions and Future Work

In this paper we have presented a novel data-driven method for aggregation in the context of natural language generation. A key aspect of our approach is the use of global inference for finding aggregations that are maximally consistent and coherent. We have formulated our inference problem as an integer linear program and shown experimentally that it outperforms a baseline clustering model by a wide margin. Beyond generation, the approach holds promise for other NLP tasks requiring the accurate partitioning of items into equivalence classes (e.g., coreference resolution).

[7] Unfortunately we cannot apply standard statistical tests such as the t-test on F-scores since they violate assumptions about underlying normal distributions. It is not possible to use an assumptions-free test like χ^2 either, since F-score is not a frequency-based measure. We can, however, use χ^2 on precision and recall, since these measures are estimated from frequency data. We thus find that the ILP model is significantly better than the clustering model on precision (χ^2 = 16.39, $p < 0.01$); the two models are not significantly different in terms of recall (χ^2 = 0.02, $p < 0.89$).

Currently, semantic grouping is carried out in our model sequentially. First, a local classifier learns the similarity of entity pairs and then ILP is employed to infer a valid partitioning. Although such a model has advantages in the face of sparse data (recall that we used a relatively small training corpus of 300 documents) and delivers good performance, it effectively decouples learning from inference. An appealing future direction lies in integrating learning and inference in a unified global framework. Such a framework would allow us to incorporate global constraints directly into the learning process.

Another important issue, not addressed in this work, is the interaction of our aggregation method with content selection and surface realization. Using an ILP formulation may be an advantage here since we could use feedback (in the form of constraints) from other components and knowlegde sources (e.g., discourse relations) to improve aggregation or indeed the generation pipeline as a whole (Marciniak and Strube, 2005).

Acknowledgments

The authors acknowledge the support of the National Science Foundation (Barzilay; CAREER grant IIS-0448168 and grant IIS-0415865) and EPSRC (Lapata; grant GR/T04540/01). Thanks to Eli Barzilay, Michael Collins, David Karger, Frank Keller, Yoong Keok Lee, Igor Malioutov, Johanna Moore, Kevin Simler, Ben Snyder, Bonnie Webber and the anonymous reviewers for helpful comments and suggestions. Any opinions, findings, and conclusions or recommendations expressed above are those of the authors and do not necessarily reflect the views of the NSF or EPSRC.

References

R. Barzilay, M. Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing*, 331–338, Vancouver.

H. Cheng, C. Mellish. 2000. Capturing the interaction between aggregation and text planning in two generation systems. In *Proceedings of the 1st International Natural Language Generation Conference*, 186–193, Mitzpe Ramon, Israel.

T. H. Cormen, C. E. Leiserson, R. L. Rivest. 1992. *Introduction to Algorithms*. The MIT Press.

H. Dalianis. 1999. Aggregation in natural language generation. *Computational Intelligence*, 15(4):384–414.

B. Di Eugenio, D. Fossati, D. Yu. 2005. Aggregation improves learning: Experiments in natural language generation for intelligent tutoring systems. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, 50–57, Ann Arbor, MI.

E. H. Hovy. 1990. Unresolved issues in paragraph planning. In R. Dale, C. Mellish, M. Zock, eds., *Current Research in Natural Language Generation*, 17–41. Academic Press, New York.

T. Marciniak, M. Strube. 2005. Beyond the pipeline: Discrete optimization in NLP. In *Proceedings of the Annual Conference on Computational Natural Language Learning*, 136–143, Ann Arbor, MI.

V. Punyakanok, D. Roth, W. Yih, D. Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of the International Conference on Computational Linguistics*, 1346–1352, Geneva, Switzerland.

M. Reape, C. Mellish. 1999. Just what is aggregation anyway? In *Proceedings of the 7th European Workshop on Natural Language Generation*, 20–29, Toulouse, France.

E. Reiter, R. Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge.

D. Roth, W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Annual Conference on Computational Natural Language Learning*, 1–8, Boston, MA.

D. Scott, C. S. de Souza. 1990. Getting the message across in RST-based text generation. In R. Dale, C. Mellish, M. Zock, eds., *Current Research in Natural Language Generation*, 47–73. Academic Press, New York.

J. Shaw. 1998. Clause aggregation using linguistic knowledge. In *Proceedings of 9th International Workshop on Natural Language Generation*, 138–147, Niagara-on-the-Lake, Ontario, Canada.

M. A. Walker, O. Rambow, M. Rogati. 2001. Spot: A trainable sentence planner. In *Proceedings of the 2nd Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, 17–24, Pittsburgh, PA.

J. Wilkinson. 1995. Aggregation in natural language generation: Another look. Technical report, Computer Science Department, University of Waterloo, 1995.

A. S. Yeung. 1999. Cognitive load and learner expertise: Split-attention and redundancy effects in reading comprehension tasks with vocabulary definitions. *Journal of Experimental Education*, 67(3):197–218.

Incorporating Speaker and Discourse Features into Speech Summarization

Gabriel Murray, Steve Renals,
Jean Carletta, Johanna Moore

University of Edinburgh, School of Informatics
Edinburgh EH8 9LW, Scotland
gabriel.murray@ed.ac.uk, s.renals@ed.ac.uk,
jeanc@inf.ed.ac.uk, j.moore@ed.ac.uk

Abstract

We have explored the usefulness of incorporating speech and discourse features in an automatic speech summarization system applied to meeting recordings from the ICSI Meetings corpus. By analyzing speaker activity, turn-taking and discourse cues, we hypothesize that such a system can outperform solely text-based methods inherited from the field of text summarization. The summarization methods are described, two evaluation methods are applied and compared, and the results clearly show that utilizing such features is advantageous and efficient. Even simple methods relying on discourse cues and speaker activity can outperform text summarization approaches.

1. Introduction

The task of summarizing spontaneous spoken dialogue from meetings presents many challenges: information is sparse; speech is disfluent and fragmented; automatic speech recognition is imperfect. However, there are numerous speech-specific characteristics to be explored and taken advantage of. Previous research on summarizing speech has concentrated on utilizing prosodic features [1, 2]. We have examined the usefulness of additional speech-specific characteristics such as discourse cues, speaker activity, and listener feedback. This speech features approach is contrasted with a second summarization approach using only textual features—a centroid method [3] using a latent semantic representation of utterances. These individual approaches are compared to a combined approach as well as random baseline summaries.

This paper also introduces a new evaluation scheme for automatic summaries of meeting recordings, using a weighted precision score based on multiple human annotations of each meeting transcript. This evaluation scheme is described in detail below and is motivated by previous findings [4] suggesting that n-gram based metrics like ROUGE [5] do not correlate well in this domain.

2. Previous Work

In the field of speech summarization in general, research investigating speech-specific characteristics has focused largely on prosodic features such as F0 mean and standard deviation, pause information, syllable duration and energy. Koumpis and Renals [1] investigated prosodic features for summarizing voicemail messages in order to send voicemail summaries to mobile devices. Hori et al. [6] have developed an integrated speech summarization approach, based on finite state transducers, in which the recognition and summarization components are composed into a single finite state transducer, reporting results on a lecture summarization task. In the Broadcast News domain, Maskey and Hirschberg [7] found that the best summarization results utilized prosodic, lexical, and structural features, while Ohtake et al. [8] explored using *only* prosodic features for summarization. Maskey and Hirschberg similarly found that prosodic features alone resulted in good quality summaries of

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 367–374,
New York, June 2006.

Broadcast News.

In the meetings domain (using the ICSI corpus), Murray et al. [2] compared text summarization approaches with feature-based approaches using prosodic features, with human judges favoring the feature-based approaches. Zechner [9] investigated summarizing several genres of speech, including spontaneous meeting speech. Though relevance detection in his work relied largely on *tf.idf* scores, Zechner also explored cross-speaker information linking and question/answer detection, so that utterances could be extracted not only according to high *tf.idf* scores, but also if they were linked to other informative utterances.

Similarly, this work aims to detect important utterances that may not be detectable according to lexical features or prosodic prominence, but are nonetheless linked to high speaker activity, decision-making, or meeting structure.

3. Summarization Approaches

The following subsections give detailed descriptions of our two summarization systems, one of which focuses on speech and discourse features while the other utilizes text summarization techniques and latent semantic analysis.

3.1. Speech and Discourse Features

In previous summarization work on the ICSI corpus [2, 4], Murray et al. explored multiple ways of applying latent semantic analysis (LSA) to a term/document matrix of weighted term frequencies from a given meeting, a development of the method in [10]. A central insight to the present work is that additional features beyond simple term frequencies can be included in the matrix before singular value decomposition (SVD) is carried out. We can use SVD to project this matrix of features to a lower dimensionality space, subsequently applying the same methods as used in [2] for extracting sentences.

The features used in these experiments included features of speaker activity, discourse cues, listener feedback, simple keyword spotting, meeting location and dialogue act length (in words).

For each dialogue act, there are features indicating which speaker spoke the dialogue act and whether the same speaker spoke the preceding and succeeding dialogue acts. Another set of features indicates how many speakers are active on either side of a given dialogue act: specifically, how many speakers were active in the preceding and succeeding five dialogue acts. To further gauge speaker activity, we located areas of high speaker interaction and indicated whether or not a given dialogue act immediately preceded this region of activity, with the motivation being that informative utterances are often provocative in eliciting responses and interaction. Additionally, we included a feature indicating which speakers most often uttered dialogue acts that preceded high levels of speaker interaction, as one way of gauging speaker status in the meeting. Another feature relating to speaker activity gives each dialogue act a score according to how active the speaker is in the meeting as a whole, based on the intuition that the most active speakers will tend to utter the most important dialogue acts.

The features for discourse cues, listener feedback, and keyword spotting were deliberately superficial, all based simply on detecting informative words. The feature for discourse cues indicates the presence or absence of words such as *decide, discuss, conclude, agree*, and fragments such as *we should* indicating a planned course of action. Listener feedback was based on the presence or absence of positive feedback cues following a given dialogue act; these include responses such as *right, exactly* and *yeah*. Keyword spotting was based on frequent words minus stopwords, indicating the presence or absence of any of the top twenty non-stopword frequent words. The discourse cues of interest were derived from a manual corpus analysis rather than being automatically detected.

A structural feature scored dialogue acts according to their position in the meeting, with dialogue acts from the middle to later portion of the meeting scoring higher and dialogue acts at the beginning and very end scoring lower. This is a feature that is well-matched to the relatively unstructured ICSI meetings, as many meetings would be expected to have informative proposals and agendas at the beginning and perhaps summary statements and conclusions at the end.

Finally, we include a dialogue act length feature motivated by the fact that informative utterances will tend to be longer than others.

The extraction method follows [11] by ranking sentences using an LSA sentence score. The

matrix of features is decomposed as follows:

$$A = USV^T$$

where U is an $m \times n$ matrix of left-singular vectors, S is an $n \times n$ diagonal matrix of singular values, and V is the $n \times n$ matrix of right-singular vectors. Using sub-matrices S and V^T, the LSA sentence scores are obtained using:

$$Sc_i^{LSA} = \sqrt{\sum_{k=1}^{n} v(i,k)^2 * \sigma(k)^2},$$

where $v(i,k)$ is the kth element of the ith sentence vector and $\sigma(k)$ is the corresponding singular value.

Experiments on a development set of 55 ICSI meetings showed that reduction to between 5–15 dimension was optimal. These development experiments also showed that weighting some features slightly higher than others resulted in much improved results; specifically, the discourse cues and listener feedback cues were weighted slightly higher.

3.2. LSA Centroid

The second summarization method is a textual approach incorporating LSA into a centroid-based system [3]. The centroid is a pseudo-document representing the important aspects of the document as a whole; in the work of [3], this pseudo-document consists of keywords and their modified *tf.idf* scores. In the present research, we take a different approach to constructing the centroid and to representing sentences in the document. First, *tf.idf* scores are calculated for all words in the meeting. Using these scores, we find the top twenty keywords and choose these as the basis for our centroid. We then perform LSA on a very large corpus of Broadcast News and ICSI data, using the Infomap tool[1]. Infomap provides a query language with which we can retrieve word vectors for our twenty keywords, and the centroid is thus represented as the average of its constituent keyword vectors [12] [13].

Dialogue acts from the meetings are represented in much the same fashion. For each dialogue act, the vectors of its constituent words are retrieved, and the dialogue act as a whole is the average of its word vectors. Extraction then proceeds by finding the dialogue act with the highest cosine similarity with the centroid, adding this to the summary, then continuing until the desired summary length is reached.

[1]http://infomap.stanford.edu

3.3. Combined

The third summarization method is simply a combination of the first two. Each system produces a ranking and a master ranking is derived from these two rankings. The hypothesis is that the strength of one system will differ from the other and that the two will complement each other and produce a good overall ranking. The first system would be expected to locate areas of high activity, decision-making, and planning, while the second would locate information-rich utterances. This exemplifies one of the challenges of summarizing meeting recordings: namely, that utterances can be important in much different ways. A comprehensive system that relies on more than one idea of importance is ideal.

4. Experimental Setup

All summaries were 350 words in length, much shorter than the compression rate used in [2] (10% of dialogue acts). The ICSI meetings themselves average around 10,000 words in length. The reasons for choosing a shorter length for summaries are that shorter summaries are more likely to be useful to a user wanting to quickly overview and browse a meeting, they present a greater summarization challenge in that the summarizer must be more exact in pinpointing the important aspects of the meeting, and shorter summaries make it more feasible to enlist human evaluators to judge the numerous summaries on various criteria in the future.

Summaries were created on both manual transcripts and speech recognizer output. The unit of extraction for these summaries was the dialogue act, and these experiments used human segmented and labeled dialogue acts rather than try to detect them automatically. In future work, we intend to incorporate dialogue act detection and labeling as part of one complete automatic summarization system.

4.1. Corpus Description

The ICSI Meetings corpus consists of 75 meetings, lasting approximately one hour each. Our test set consists of six meetings, each with multiple human annotations. Annotators were given access to a graphical user interface (GUI) for browsing an individual meeting that included earlier human annotations: an orthographic transcription time-synchronized with the audio, and a topic segmentation based on a shallow hierarchical decomposition with keyword-based text labels describing each topic segment. The annotators were told to construct a textual summary of the meeting aimed at someone who is interested in the research being carried out, such as a researcher who does similar work elsewhere, using four headings:

- general abstract: "why are they meeting and what do they talk about?";
- decisions made by the group;
- progress and achievements;
- problems described

The annotators were given a 200 word limit for each heading, and told that there must be text for the general abstract, but that the other headings may have null annotations for some meetings. Annotators who were new to the data were encouraged to listen to a meeting straight through before beginning to author the summary.

Immediately after authoring a textual summary, annotators were asked to create an extractive summary, using a different GUI. This GUI showed both their textual summary and the orthographic transcription, without topic segmentation but with one line per dialogue act based on the pre-existing MRDA coding [14]. Annotators were told to extract dialogue acts that together would convey the information in the textual summary, and could be used to support the correctness of that summary. They were given no specific instructions about the number or percentage of acts to extract or about redundant dialogue acts. For each dialogue act extracted, they were then required in a second pass to choose the sentences from the textual summary supported by the dialogue act, creating a many-to-many mapping between the recording and the textual summary. Although the expectation was that each extracted dialogue act and each summary sentence would be linked to something in the opposing resource, we told the annotators that under some circumstances dialogue acts and summary sentences could stand alone.

We created summaries using both manual transcripts as well as automatic speech recognition (ASR) output. The AMI-ASR system [15] is described in more detail in [4] and the average word error rate (WER) for the corpus is 29.5%.

4.2. Evaluation Frameworks

The many-to-many mapping of dialogue acts to summary sentences described in the previous section allows us to evaluate our extractive summaries according to how often each annotator linked a given extracted dialogue act to a summary sentence. This is somewhat analogous to Pyramid weighting [16], but with dialogue acts as the SCUs. In fact, we can calculate weighted precision, recall and f-score using these annotations, but because the summaries created are so short, we focus on weighted precision as our central metric. For each dialogue act that the summarizer extracts, we count the number of times that each annotator links that dialogue act to a summary sentence. For a given dialogue act, it may be that one annotator links it 0 times, one annotator links it 1 time, and the third annotator links it two times, resulting in an average score of 1 for that dialogue act. The scores for all of the summary dialogue acts can be calculated and averaged to create an overall summary score.

ROUGE scores, based on n-gram overlap between human abstracts and automatic extracts, were also calculated for comparison [5]. ROUGE-2, based on bigram overlap, is considered the most stable as far as correlating with human judgments, and this was therefore our ROUGE metric of interest. ROUGE-SU4, which evaluates bigrams with intervening material between the two elements of the bigram, has recently been shown in the context of the Document Understanding Conference (DUC)[2] to bring no significant additional information as compared with ROUGE-2. Results from [4] and from DUC 2005 also show that ROUGE does not always correlate well with human judgments. It is therefore included in this research in the hope of further determining how reliable the

[2] http://duc.nist.gov

ROUGE metric is for our domain of meeting summarization.

5. Results

The experimental results are shown in figure 1 (weighted precision) and figure 2 (ROUGE-2) and are discussed below.

5.1. Weighted Precision Results

For weighted precision, the speech features approach was easily the best and scored significantly better than the centroid and random approaches (ANOVA,$p<0.05$), attaining an averaged weighted precision of 0.52. The combined approach did not improve upon the speech features approach but was not significantly worse either. The randomly created summaries scored much lower than all three systems.

The superior performance of the speech features approach compared to the LSA centroid method closely mirrors results on the ICSI development set, where the centroid method scored 0.23 and the speech features approach scored 0.42. For the speech features approach on the test set, the best feature by far was dialogue act length. Removing this feature resulted in the precision score being nearly halved. This mirrors results from Maskey and Hirschberg [7], who found that the length of a sentence in seconds and its length in words were the two best features for predicting summary sentences. Both the simple keyword spotting and the discourse cue detection features caused a lesser decline in precision when removed, while other features of speaker activity had a negligible impact on the test results.

Interestingly, the weighted precision scores on ASR were not significantly worse for any of the summarization approaches. In fact, the centroid approach scored very slightly higher on ASR output than on manual transcripts. In [17] and [2] it was similarly found that summarizing with ASR output did not cause great deterioration in the quality of the summaries. It is not especially surprising that the speech features approach performed similarly on both manual and ASR transcripts, as many of its features based on speaker exchanges and speaker activity would be unaffected by ASR errors. The speech features approach is still significantly better than the random and centroid summaries, and is not significantly better than the combined approach on ASR.

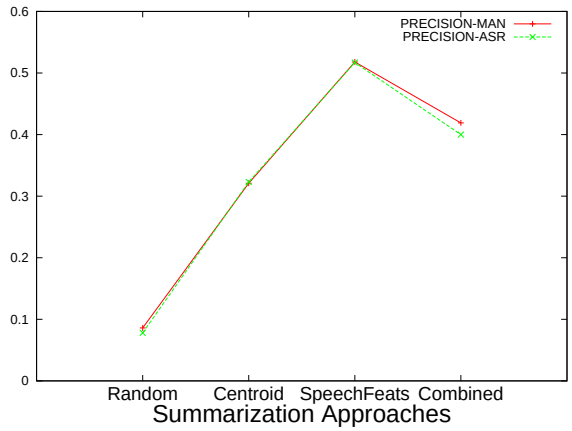


Figure 1: *Weighted Precision Results on Test Set*

5.2. ROUGE Results

The ROUGE results greatly differed from the weighted precision results in several ways. First, the centroid method was considered to be the best, with a ROUGE-2 score of 0.047 compared with 0.041 for the speech features approach. Second, there were not as great of differences between the four systems according to ROUGE as there were according to weighted precision. In fact, the random summaries of manual transcripts are not significantly worse than the other approaches, according to ROUGE-2. Neither the combined approach nor the speech features approach is significantly worse than the centroid system, with the combined approach generally scoring on par with the centroid scores.

The third difference relates to summarization on ASR output. ROUGE-2 has the random system and the combined system showing sharp declines when applied to ASR transcripts. The speech features and centroid approaches do not show declines. Random summaries are significantly worse than both the centroid summaries ($p<0.1$) and speech features summaries ($p<0.05$). Though the combined approach declines on ASR output, it is not significantly worse than the other systems.

To get an idea of a ROUGE-2 upper bound, for each meeting in the test set we left one human abstract out and compared it with the remaining abstracts. The result was an average ROUGE-2 score of .086.

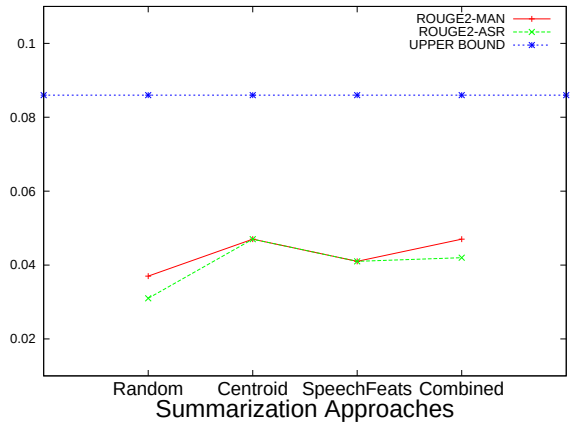


Figure 2: *ROUGE-2 Results on Test Set*

ROUGE-1 and ROUGE-SU4 show no significant differences between the centroid and speech features approaches.

5.3. Correlations

There is no significant correlation between macroaveraged ROUGE and weighted precision scores across the meeting set, on both ASR and manual transcripts. The Pearson correlation is 0.562 with a significance of $p < 0.147$. The Spearman correlation is 0.282 with a significance of $p < 0.498$. The correlation of scores across each test meeting is worse yet, with a Pearson correlation of 0.185 ($p<0.208$) and a Spearman correlation of 0.181 ($p<0.271$).

5.4. Sample Summary

The following is the text of a summary of meeting Bed004 using the speech features approach:

-so its possible that we could do something like a summary node of some sort that

-and then the question would be if if those are the things that you care about uh can you make a relatively compact way of getting from the various inputs to the things you care about

-this is sort of th the second version and i i i look at this maybe just as a you know a a whatever uml diagram or you know as just a uh screen shot not really as a bayes net as john johno said

-and um this is about as much as we can do if we dont w if we want to avoid uh uh a huge combinatorial explosion where we specify ok if its this and this but that is not the case and so forth it just gets really really messy

-also it strikes me that we we m may want to approach the point where we can sort of try to find a uh a specification for some interface here that um takes the normal m three l looks at it

-so what youre trying to get out of this deep co cognitive linguistics is the fact that w if you know about source source paths and goals and nnn all this sort of stuff that a lot of this is the same for different tasks

-what youd really like of course is the same thing youd always like which is that you have um a kind of intermediate representation which looks the same o over a bunch of inputs and a bunch of outputs

-and pushing it one step further when you get to construction grammar and stuff what youd like to be able to do is say you have this parser which is much fancier than the parser that comes with uh smartkom

-in independent of whether it about what is this or where is it or something that you could tell from the construction you could pull out deep semantic information which youre gonna use in a general way

6. Discussion

Though the speech features approach was considered the best system, it is unclear why the combined approach did not yield improvement. One possibility relates to the extreme brevity of the summaries: because the summaries are only 350 words in length, it is possible to have two summaries of the same meeting which are equally good but completely non-overlapping in content. In other words, they both extract informative dialogue acts, but not the same ones. Combining the rankings of two such systems might create a third system which is comparable but not any better than either of the first two systems alone. However, it is still possible that the combined system will be better in terms of balancing the two types of importance discussed above: utterances that contain a lot of informative content and keywords and utterances that relate to decision-making and meeting structure.

ROUGE did not correlate well with the weighted precision scores, a result that adds to the previous evidence that this metric may not be reliable in the domain of meeting summarization.

It is very encouraging that the summarization approaches in general seem immune to the WER of the ASR output. This confirms previous findings such as [17] and [2], and the speech and structural features used herein are particularly unaffected by a moderately high WER. The reason for the random summarizaton system not suffering

a sharp decline when applied to ASR may be due to the fact that its scores were already so low that it couldn't deteriorate any further.

7. Future Work

The above results show that even a relatively small set of speech, discourse, and structural features can outperform a text summarization approach on this data, and there are many additional features to be explored. Of particular interest to us are features relating to speaker status, i.e. features that help us determine who is leading the meeting and who it is that others are deferring to. We would also like to more closely investigate the relationship between areas of high speaker activity and informative utterances.

In the immediate future, we will incorporate these features into a machine-learning framework, building support vector models trained on the extracted and non-extracted classes of the training set.

Finally, we will apply these methods to the AMI corpus [18] and create summaries of comparable length for that meeting set. There are likely to be differences regarding usefulness of certain features due to the ICSI meetings being relatively unstructured and informal and the AMI hub meetings being more structured with a higher information density.

8. Conclusion

The results presented above show that using features related to speaker activity, listener feedback, discourse cues and dialogue act length can outperform the lexical methods of text summarization approaches. More specifically, the fact that there are multiple types of important utterances requires that we use multiple methods of detecting importance. Lexical methods and prosodic features are not necessarily going to detect utterances that are relevant to agreement, decision-making or speaker activity. This research also provides further evidence that ROUGE does not correlate well with human judgments in this domain. Finally, it has been demonstrated that high WER for ASR output does not significantly decrease summarization quality.

9. Acknowledgements

Thanks to Thomas Hain and the AMI-ASR group for speech recognition output. This work was partly supported by the European Union 6th FWP IST Integrated Project AMI (Augmented Multiparty Interaction, FP6-506811, publication AMI-150).

10. References

[1] K. Koumpis and S. Renals, "Automatic summarization of voicemail messages using lexical and prosodic features," *ACM Transactions on Speech and Language Processing*, vol. 2, pp. 1–24, 2005.

[2] G. Murray, S. Renals, and J. Carletta, "Extractive summarization of meeting recordings," in *Proceedings of the 9th European Conference on Speech Communication and Technology, Lisbon, Portugal*, September 2005.

[3] D. Radev, S. Blair-Goldensohn, and Z. Zhang, "Experiments in single and multidocument summarization using mead," in *The Proceedings of the First Document Understanding Conference, New Orleans, LA*, September 2001.

[4] G. Murray, S. Renals, J. Carletta, and J. Moore, "Evaluating automatic summaries of meeting recordings," in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, Workshop on Machine Translation and Summarization Evaluation (MTSE), Ann Arbor, MI, USA*, June 2005.

[5] C.-Y. Lin and E. H. Hovy, "Automatic evaluation of summaries using n-gram co-occurrence statistics," in *Proceedings of HLT-NAACL 2003, Edmonton, Calgary, Canada*, May 2003.

[6] T. Hori, C. Hori, and Y. Minami, "Speech summarization using weighted finite-state transducers," in *Proceedings of the 8th European Conference on Speech Communication and Technology, Geneva, Switzerland*, September 2003.

[7] S. Maskey and J. Hirschberg, "Comparing lexial, acoustic/prosodic, discourse and structural features for speech summarization," in *Proceedings of the 9th European Conference on Speech Communication and Technology, Lisbon, Portugal*, September 2005.

[8] K. Ohtake, K. Yamamoto, Y. Toma, S. Sado, S. Masuyama, and S. Nakagawa, "Newscast speech summarization via sentence shortening based on prosodic features," in *Proceedings of the ISCA and IEEE Workshop on Spontaneous Speech Processing and Recognition, Tokyo, Japan*, April 2003,.

[9] K. Zechner, "Automatic summarization of open-domain multiparty dialogues in diverse genres," *Computational Linguistics*, vol. 28, no. 4, pp. 447–485, 2002.

[10] Y. Gong and X. Liu, "Generic text summarization using relevance measure and latent semantic analysis," in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New Orleans, Louisiana, USA*, September 2001, pp. 19–25.

[11] J. Steinberger and K. Ježek, "Using latent semantic analysis in text summarization and summary evaluation," in *Proceedings of ISIM 2004, Roznov pod Radhostem, Czech Republic*, April 2004, pp. 93–100.

[12] P. Foltz, W. Kintsch, and T. Landauer, "The measurement of textual coherence with latent semantic analysis," *Discourse Processes*, vol. 25, 1998.

[13] B. Hachey, G. Murray, and D. Reitter, "The embra system at duc 2005: Query-oriented multi-document summarization with a very large latent semantic space," in *Proceedings of the Document Understanding Conference (DUC) 2005, Vancouver, BC, Canada*, October 2005.

[14] E. Shriberg, R. Dhillon, S. Bhagat, J. Ang, , and H. Carvey, "The ICSI meeting recorder dialog act (MRDA) corpus," in *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue, Cambridge, MA, USA*, April-May 2004, pp. 97–100.

[15] T. Hain, J. Dines, G. Garau, M. Karafiat, D. Moore, V. Wan, R. Ordelman, I.Mc.Cowan, J.Vepa, and S.Renals, "An investigation into transcription of conference room meetings," *Proceedings of the 9th European Conference on Speech Communication and Technology, Lisbon, Portugal*, September 2005.

[16] A. Nenkova and B. Passonneau, "Evaluating content selection in summarization: The pyramid method," in *Proceedings of HLT-NAACL 2004, Boston, MA, USA*, May 2004.

[17] R. Valenza, T. Robinson, M. Hickey, and R. Tucker, "Summarization of spoken audio through information extraction," in *Proceedings of the ESCA Workshop on Accessing Information in Spoken Audio, Cambridge UK*, April 1999, pp. 111–116.

[18] J. Carletta, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, W. Kraaij, M. Kronenthal, G. Lathoud, M. Lincoln, A. Lisowska, I. McCowan, W. Post, D. Reidsma, and P. Wellner, "The AMI meeting corpus: A pre-announcement," in *Proceedings of MLMI 2005, Edinburgh, UK*, June 2005.

Nuggeteer: Automatic Nugget-Based Evaluation using Descriptions and Judgements

Gregory Marton **Alexey Radul**
Infolab Group, MIT CSAIL
Cambridge, MA 02139
{gremio,axch}@mit.edu

Abstract

The TREC Definition and Relationship questions are evaluated on the basis of information nuggets that may be contained in system responses. Human evaluators provide informal descriptions of each nugget, and judgements (assignments of nuggets to responses) for each response submitted by participants. While human evaluation is the most accurate way to compare systems, approximate automatic evaluation becomes critical during system development.

We present Nuggeteer, a new automatic evaluation tool for nugget-based tasks. Like the first such tool, Pourpre, Nuggeteer uses words in common between candidate answer and answer key to approximate human judgements. Unlike Pourpre, but like human assessors, Nuggeteer creates a judgement for each candidate-nugget pair, and can use existing judgements instead of guessing. This creates a more readily interpretable aggregate score, and allows developers to track individual nuggets through the variants of their system. Nuggeteer is quantitatively comparable in performance to Pourpre, and provides qualitatively better feedback to developers.

1 Introduction

The TREC Definition and Relationship questions are evaluated on the basis of information nuggets, abstract pieces of knowledge that, taken together, comprise an answer. Nuggets are described informally, with abbreviations, misspellings, etc., and each is associated with an importance judgement: 'vital' or 'okay'.[1] In some sense, nuggets are like WordNet synsets, and their descriptions are like glosses. Responses may contain more than one nugget—when they contain more than one piece of knowledge from the answer. The median scores of today's systems are frequently zero; most responses contain no nuggets (Voorhees, 2005).

Human assessors decide what nuggets make up an answer based on some initial research and on pools of top system responses for each question. Answer keys list, for each nugget, its id, importance, and description; two example answer keys are shown in Figures 1 and 2. Assessors make binary decisions about each response, whether it contains each nugget. When multiple responses contain a nugget, the assessor gives credit only to the (subjectively) best response.

Using the judgements of the assessors, the final score combines the recall of the available vital nuggets, and the length (discounting whitespace) of the system response as a proxy for precision. Nuggets valued 'okay' contribute to precision by increasing the length allowance, but do not contribute to recall. The scoring formula is shown in Figure 3.

[1] Nuggeteer implements the *pyramid* scoring system from (Lin and Demner-Fushman, 2006), designed to soften the dis-

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 375–382,
New York, June 2006.

Qid 87.8: 'other' question for target Enrico Fermi

1	*vital*	belived in partical's existence and named it neutrino
2	*vital*	Called the atomic Bomb an evil thing
3	*okay*	Achieved the first controlled nuclear chain reaction
4	*vital*	Designed and built the first nuclear reactor
5	*okay*	Concluded that the atmosphere was in no real danger before Trinity test
6	*okay*	co-developer of the atomic bomb
7	*okay*	pointed out that the galaxy is 100,000 light years across

Figure 1: The "answer key" to an "other" question from 2005.

The analyst is looking for links between Colombian businessmen and paramilitary forces. Specifically, the analyst would like to know of evidence that business interests in Colombia are still funding the AUC paramilitary organization.

1	*vital*	Commander of the national paramilitary umbrella organization claimed his group enjoys growing support from local and international businesses
2	*vital*	Columbia's Chief prosecutor said he had a list of businessmen who supported right-wing paramilitary squads and warned that financing outlawed groups is a criminal offense
3	*okay*	some landowners support AUC for protections services
4	*vital*	Rightist militias waging a dirty war against suspected leftists in Colombia enjoy growing support from private businessmen
5	*okay*	The AUC makes money by taxing Colombia's drug trade
6	*okay*	The ACU is estimated to have 6000 combatants and has links to government security forces.
7	*okay*	Many ACU fighters are former government soldiers

Figure 2: The "answer key" to a relationship question.

Let

r	# of *vital* nuggets returned in a response
a	# of *okay* nuggets returned in a response
R	# of *vital* nuggets in the answer key
l	# of non-whitespace characters in the entire answer string

Then

$$\text{"recall" } \mathcal{R} = r/R$$

$$\text{"allowance" } \alpha = 100 \times (r + a)$$

$$\text{"precision" } \mathcal{P} = \begin{cases} 1 & \text{if } l < \alpha \\ 1 - \frac{l-\alpha}{l} & \text{otherwise} \end{cases}$$

Finally, the $F(\beta) = \dfrac{(\beta^2 + 1) \times \mathcal{P} \times \mathcal{R}}{\beta^2 \times \mathcal{P} + \mathcal{R}}$

Figure 3: Official definition of F-measure.

Automatic evaluation of systems is highly desirable. Developers need to know whether one system performs better or worse than another. Ideally, they would like to know which nuggets were lost or gained. Because there is no exhaustive list of snippets from the document collection that contain each nugget, an exact automatic solution is out of reach. Manual evaluation of system responses is too time consuming to be effective for a development cycle.

The Qaviar system first described an approximate automatic evaluation technique using keywords, and Pourpre was the first publicly available implementation for these nugget-based tasks. (Breck et al., 2000; Lin and Demner-Fushman, 2005). Pourpre calculates an idf- or count-based, stemmed, unigram similarity between each nugget description and each

tinction between 'vital' and 'okay'.

candidate system response. If this similarity passes a threshold, then it uses this similarity to assign a partial value for recall and a partial length allowance, reflecting the uncertainty of the automatic judgement. Importantly, it yields a ranking of systems very similar to the official ranking (See Table 2).

Nuggeteer offers three important improvements:

- interpretability of the scores, as compared to official scores,
- use of known judgements for exact information about some responses, and
- information about individual nuggets, for detailed error analysis.

Nuggeteer makes scores interpretable by making binary decisions about each nugget and each system response, just as assessors do, and then calculating the final score in the usual way. We will show that Nuggeteer's absolute error is comparable to human error, and that the 95% confidence intervals Nuggeteer reports are correct around 95% of the time.

Nuggeteer assumes that if a system response was ever judged by a human assessor to contain a particular nugget, then other identical responses also contain that nugget. When this is not true among the human judgements, we claim it is due to annotator error. This assumption allows developers to add their own judgements and have the responses they've adjudicated scored "exactly" by Nuggeteer.

These features empower developers to track not only the numeric value of a change to their system, but also its effect on retrieval of each nugget.

2 Approach

Nuggeteer builds one binary classifier per nugget for each question, based on n-grams (up to trigrams) in the description and optionally in any provided judgement files. The classifiers use a weight for each n-gram, an informativeness measure for each n-gram, and a threshold for accepting a response as bearing the nugget.

2.1 N-gram weight

The idf-based weight for an n-gram $w_1...w_n$ is the sum of unigram idf counts from the AQUAINT corpus of English newspaper text, the corpus from which responses for the TREC tasks are drawn. We did not explore using n-gram idfs. A tf component is not meaningful because the data are so sparse.

2.2 Informativeness

Let G be the set of nuggets for some question. Informativeness of an n-gram for a nugget g is calculated based on how many other nuggets in that question ($\in G$) contain the n-gram. Let

$$i(g, w_1...w_n) = \begin{cases} 1 & \text{if } count(g, w_1..w_n) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $count(g, w_1...w_n)$ is the number of occurrences of the n-gram in responses containing the nugget g.

Then informativeness is:

$$I(g, w_1...w_n) = 1 - \frac{\sum_{g' \in G} i(g', w_1...w_n)}{|G|} \quad (2)$$

This captures the Bayesian intuition that the more outcomes a piece of evidence is associated with, the less confidence we can have in predicting the outcome based on that evidence.

2.3 Judgement

Nuggeteer does not guess on responses which have been judged by a human to contain a nugget, or those which have unambiguously judged not to, but assigns the known judgement.[2]

For unseen responses, we determine the n-gram recall for each nugget g and candidate response $w_1...w_l$ by breaking the candidate into n-grams and finding the sum of scores:

$$Recall(g, w_1...w_l) = \sum_{k=0}^{n-1} \sum_{i=0}^{l-k} W(g, w_i...w_{i+k}) * I(g, w_i...w_{i+k}) \quad (3)$$

The candidate is considered to contain all nuggets whose recall exceeds some threshold. Put another

[2] If a response was submitted, and no response from the same system was judged to contain a nugget, then the response is considered to not contain the nugget. We normalized whitespace and case for matching previously seen responses.

way, we build an n-gram language model for each nugget, and assign those nuggets whose predicted likelihood exceeds a threshold.

When several responses contain a nugget, Nuggeteer picks the *first* (instead of the best, as assessors can) for purposes of scoring.

2.4 Parameter Estimation

We explored a number of parameters in the scoring function: stemming, n-gram size, idf weights vs. count weights, and the effect of removing stopwords. We tested all 24 combinations, and for each experiment, we cross-validated by leaving out one submitted system, or where possible, one submitting institution (to avoid training and testing on potentially very similar systems).[3]

Each experiment was performed using a range of thresholds for Equation 3 above, and we selected the best performing threshold for each data set.[4] Because the threshold was selected after cross-validation, it is exposed to overtraining. We used a single global threshold to minimize this risk, but we have no reason to think that the thresholds for different nuggets are related.

Selecting thresholds as part of the training process can maximize accuracy while eliminating overtraining. We therefore explored Bayesian models for automatic threshold selection. We model assignment of nuggets to responses as caused by the scores according to a noisy threshold function, with separate false positive and false negative error rates. We varied thresholds and error rates by entire dataset, by question, or by individual nugget, evaluating them using Bayesian model selection.

3 The Data

For our experiments, we used the definition questions from TREC2003, the 'other' questions from TREC2004 and TREC2005, and the relationship questions from TREC2005. (Voorhees, 2003; Voorhees, 2004; Voorhees, 2005) The distribution of nuggets and questions is shown for each data set in Table 1. The number of nuggets by number of system responses assigned that nugget (difficulty of nuggets, in a sense) is shown in Figure 4. More than a quarter of relationship nuggets were not found by any system. Among all data sets, many nuggets were found in none or just a few responses.

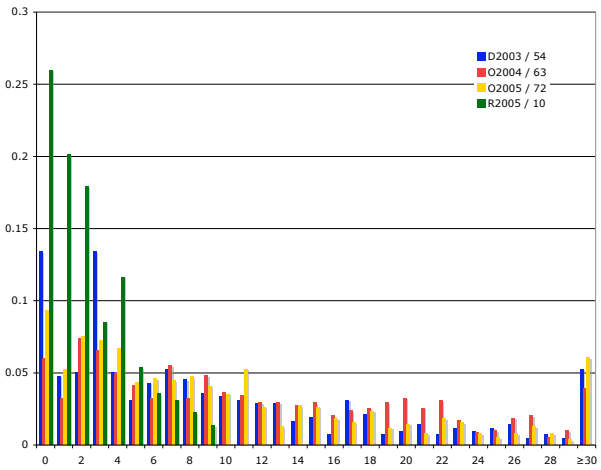


Figure 4: Percents of nuggets, binned by the number of systems that found each nugget.

4 Results

We report correlation (R^2), and Kendall's τ_b, following Lin and Demner-Fushman. Nuggeteer's scores are in the same range as real system scores, so we also report average root mean squared error from the official results. We 'corrected' the official judgements by assigning a nugget to a response if that response was judged to contain that nugget in any assessment for any system.

4.1 Comparison with Pourpre

(Lin et al., 2005) report Pourpre and Rouge performance with Pourpre optimal thresholds for TREC definition questions, as reproduced in Table 2. Nuggeteer's results are shown in the last column.[5]

Table 3 shows a comparison of Pourpre and Nuggeteer's correlations with official scores. As ex-

[3]For TREC2003 and TREC2004, the run-tags indicate the submitting institution. For TREC2005 we did not run the non-anonymized data in time for this submission. In the TREC2005 Relationship task, RUN-1 was withdrawn.

[4]Thresholds for Pourpre were also selected this way.

[5]We report only micro-averaged results, because we wish to emphasize the interpretability of Nuggeteer scores. While the correlations of macro-averaged scores with official scores may be higher (as seems to be the case for Pourpre), the actual values of the micro-averaged scores are more interpretable because they include a variance.

	#ques	#vital	#okay	#n/q	#sys	#r/s	#r/q/s
D 2003:	50	207	210	9.3± 1.0	54	526± 180	10.5± 1.2
O 2004:	64	234	346	10.1± .7	63	870± 335	13.6± 0.9
O 2005:	75	308	450	11.1± .6	72	1277± 260[a]	17.0± 0.6[a]
R 2005:	25	87	136	9.9± 1.6	10	379± 222[b]	15.2± 1.6[b]
[a] excluding RUN-135: 410,080 responses							5468 ± 5320
[b] excluding RUN-7: 6436 responses							257 ± 135

Table 1: For each data set (D="definition", O="other", R="relationship"), the number of questions, the numbers of vital and okay nuggets, the average total number of nuggets per question, the number of participating systems, the average number of responses per system, and the average number of responses per question over all systems.

	POURPRE				ROUGE		NUGGETEER
Run	micro, cnt	macro, cnt	micro, *idf*	macro, *idf*	default	stop	nostem, bigram, micro, idf
D 2003 ($\beta = 3$)	0.846	**0.886**	0.848	0.876	0.780	**0.816**	**0.879**
D 2003 ($\beta = 5$)	0.890	**0.878**	0.859	0.875	0.807	**0.843**	**0.849**
O 2004 ($\beta = 3$)	0.785	**0.833**	0.806	0.812	0.780	**0.786**	**0.898**
O 2005 ($\beta = 3$)	0.598	**0.709**	0.679	0.698	0.662	**0.670**	**0.858**
R 2005 ($\beta = 3$)		**0.697**					**1**

Table 2: Kendall's τ correlation between rankings generated by POURPRE/ROUGE/NUGGETEER and official scores, for each data set (D="definition", O="other", R="relationship"). τ=1 means same order, τ=-1 means reverse order. Pourpre and Rouge scores reproduced from (Lin and Demner-Fushman, 2005).

	POURPRE	NUGGETEER	
Run	R^2	R^2	$\sqrt{mse}$
D 2003 ($\beta = 3$)	0.963	0.966	0.067
D 2003 ($\beta = 5$)	0.965	0.971	0.077
O 2004 ($\beta = 3$)	0.929	0.982	0.026
O 2005 ($\beta = 3$)	0.916	0.952	0.026
R 2005 ($\beta = 3$)	0.764	0.993	0.009

Table 3: Correlation (R^2) and Root Mean Squared Error ($\sqrt{mse}$) between scores generated by Pourpre/Nuggeteer and official scores, for the same settings as the τ comparison above.

pected from the Kendall's τ comparisons, Pourpre's correlation is about the same or higher in 2003, but fares progressively worse in the subsequent tasks.

To ensure that Pourpre scores correlated sufficiently with official scores, Lin and Demner-Fushman used the difference in official score between runs whose ranks Pourpre had swapped, and showed that the majority of swaps were between runs whose official scores were less than the 0.1 apart, a threshold for assessor agreement reported in (Voorhees, 2003).

Nuggeteer scores are not only correlated with, but actually meant to approximate, the assessment scores; thus we can use a stronger evaluation: root mean squared error of Nuggeteer scores against official scores. This estimates the average difference between the Nuggeteer score and the official score, and at 0.077, the estimate is below the 0.1 threshold. This evaluation is meant to show that the scores are "good enough" for experimental evaluation, and in Section 4.4 we will substantiate Lin and Demner-Fushman's observation that higher correlation scores may reflect overtraining rather than actual improvement.

Accordingly, rather than reporting the best Nuggeteer scores (Kendall's τ and R^2) above, we follow Pourpre's lead in reporting a single variant (no stemming, bigrams) that performs well across the data sets. As with Pourpre's evaluation, the par-

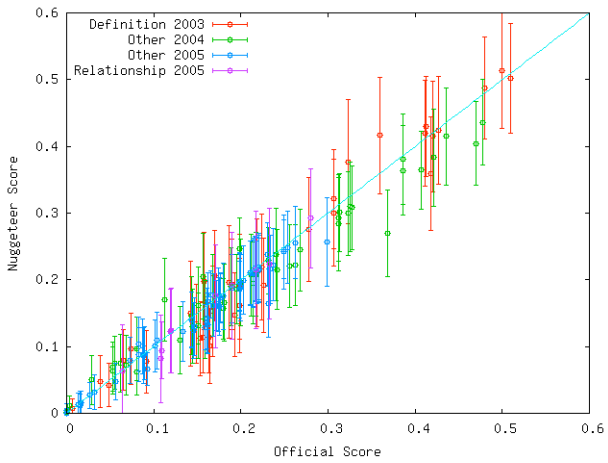


Figure 5: Scatter graph of official scores plotted against Nuggeteer scores (idf term weighting, no stemming, bigrams) for each data set (all F-measures have $\beta = 3$), with the Nuggeteer 95% confidence intervals on the score. Across the four datasets, 6 systems (3%) have an official score outside Nuggeteer's 95% confidence interval.

ticular thresholds for each year are experimentally optimized. A scatter plot of Nuggeteer performance on the definition tasks is shown in Figure 5.

4.2 N-gram size and stemming

A hypothesis advanced with Pourpre is that bigrams, trigrams, and longer n-grams will primarily account for the fluency of an answer, rather than its semantic content, and thus not aid the scoring process. We included the option to use longer n-grams within Nuggeteer, and have found that using bigrams can yield very slightly better results than using unigrams. From inspection, bigrams sometimes capture named entity and grammatical order features.

Experiments with Pourpre showed that stemming hurt slightly at peak performances. Nuggeteer has the same tendency at all n-gram sizes.

Figure 6 compares Kendall's τ over the possible thresholds, n-gram lengths, and stemming. The choice of threshold matters by far the most.

4.3 Term weighting and stopwords

Removing stopwords or giving unit weight to all terms rather than an idf-based weight made no substantial difference in Nuggeteer's performance.

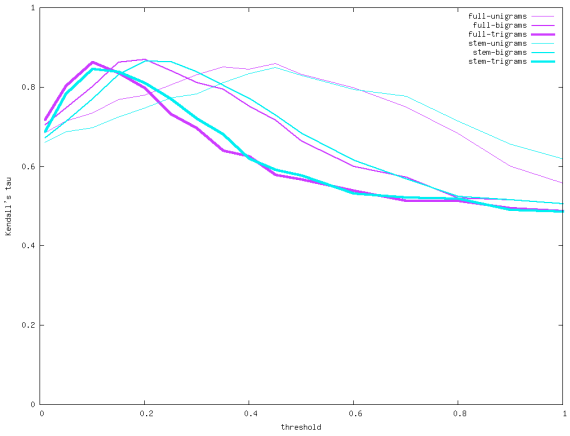


Figure 6: Fixed thresholds vs. Kendall's τ for unigrams, bigrams, or trigrams averaged over the three years of definition data using $F(\beta = 3)$.

Model	$\log_{10} P(\text{Data}\|\text{Model})$
optimally biased coin	-2780
global threshold	-2239
per-question thresholds	-1977
per-nugget thresholds	**-1546**
per-nugget errors and thr.	-1595

Table 4: The probabilities of the data given several models: a baseline coin, three models of different granularity with globally specified false positive and negative error rates, and a model with too many parameters, where even the error rates have per-nugget granularity. We select the most probable model, the per-nugget threshold model.

4.4 Thresholds

We experimented with Bayesian models for automatic threshold selection. In the models, a system response contains or does not contain each nugget as a function of the response's Nuggeteer score plus noise. Table 4 shows that, as expected, the best models do not make assumptions about thresholds being equal within a question or dataset. It is interesting to note that Bayesian inference catches the overparametrization of the model where error rates vary per-nugget as well. In essence, we do not need those additional parameters to explain the variation in the data.

The τ of the best selection of parameters on the 2003 data set using the model with one threshold per

nugget and global errors is 0.837 ($\sqrt{mse}$=0.037). We have indeed overtrained the best threshold for this dataset (compare τ=0.879, $\sqrt{mse}$=0.067 in Tables 2 and 3), suggesting that the numeric differences in Kendall's Tau shown between the Nuggeteer, Pourpre, and Rouge systems are not indicative of true performance. The Bayesian model promises settings free of overtraining, and thus more accurate judgements in terms of $\sqrt{mse}$ and individual nugget classification accuracy.

4.5 Training on System Responses

Intuitively, if a fact is expressed by a system response, then another response with similar n-grams may also contain the same fact. To test this intuition, we tried expanding our judgement method (Equation 3) to select the maximum judgement score from among those of the nugget description and each of the system responses judged to contain that nugget.

Unfortunately, the assessors did not mark which *portion* of a response expresses a nugget, so we also find spurious similarity, as shown in Figure 7. The final results are not conclusively better or worse overall, and the process is far more expensive.

We are currently exploring the same extension for multiple "nugget descriptions" generated by manually selecting the appropriate portions of system responses containing each nugget.

4.6 Judgment Precision and Recall

Because Nuggeteer makes a nugget classification for each system response, we can report precision and recall on the nugget assignments. Table 5 shows Nuggeteer's agreement rate with assessors on whether each response contains a nugget. [6]

4.7 Novel Judgements

Approximate evaluation will tend to undervalue new results, simply because they may not have keyword overlap with existing nugget descriptions. We are therefore creating tools to help developers manually assess their system outputs.

As a proof of concept, we ran Nuggeteer on the best 2005 "other" system (not giving Nuggeteer the official judgements), and manualy corrected its guesses.[7] Assessment took about 6 hours, and our judgements had precision of 78% and recall of 90%, for F-measure 0.803± 0.065 (compare Table 5). The official score of .299 was still within the confidence interval, but now on the high side rather than the low (.257± .07), because we found the answers quite good. In fact, we were often tempted to add new nuggets! We later learned that it was a manual run, produced by a student at the University of Maryland.

Data set	**best** F($\beta = 1$)	**default** F($\beta = 1$)
2003 defn	0.68± .01	0.66± .02
2004 other	0.73± .01	0.70± .01
2005 other	0.87± .01	0.86± .01
2005 reln	0.75± .04	0.72± .05

Table 5: Nuggeteer agreement with official judgements, under best settings for each year, and under the default settings.

5 Discussion

Pourpre pioneered automatic nugget-based assessment for definition questions, and thus enabled a rapid experimental cycle of system development. Nuggeteer improves on that functionality, and critically adds:

- an interpretable score, comparable to official scores, with near-human error rates,
- a reliable confidence interval on the estimated score,
- scoring known responses exactly,
- support for improving the accuracy of the score through additional annotation, and
- a more robust training process

We have shown that Nuggeteer evaluates the definition and relationship tasks with comparable rank swap rates to Pourpre. We explored the effects of stemming, term weighting, n-gram size, stopword removal, and use of system responses for training, all with little effect. We showed that previous methods of selecting a threshold overtrained, and have

[6] Unlike human assessors, Nuggeteer is not able to pick the "*best*" response containing a nugget if multiple responses have it, and will instead pick the *first*, so these values are artifactually low. However, 2005 results may be high because these results reflect anonymized runs.

[7] We used a low threshold to make the task mostly correcting and less searching. This is clearly not how assessors should work, but is expedient for developers.

question id 1901, *response rank* 2, *response score* 0.14

response text: `best american classical music bears its stamp: witness aaron copland, whose "american-sounding" music was composed by a`
(the response was a sentence fragment)

assigned nugget description: born brooklyn ny 1900

bigram matches: "american classical", "american-sounding music", "best american", "whose american-sounding", "witness aaron", "copland whose", "stamp witness", ...

response containing the nugget: `Even the best American classical music bears its stamp: witness Aaron Copland, whose ``American-sounding'' music was composed by a Brooklyn-born Jew of Russian lineage who studied in France and salted his scores with jazz-derived syncopations, Mexican folk tunes and cowboy ballads.` NYT19981210.0106

Figure 7: This answer to the definition question on Aaron Copeland is assigned the nugget "born brooklyn ny 1900" at a recall score well above that of the background, despite containing none of those words.

briefly described a promising way to select finer-grained thresholds automatically.

Our experiences in using judgements of system responses point to the need for a better annotation of nugget content. It is possible to give Nuggeteer multiple nugget descriptions for each nugget. Manually extracting the relevant portions of correctly-judged system responses may not be an overly arduous task, and may offer higher accuracy. It would be ideal if the community—including the assessors—were able to create and promulgate a gold-standard set of nugget descriptions for previous years.

Nuggeteer currently supports evaluation for the TREC definition, 'other', and relationship tasks, for the AQUAINT opinion pilot [8], and is under development for the DARPA GALE task [9].

6 Acknowledgements

We would like to thank Jimmy Lin and Dina Demner-Fushman for valuable discussions, for Figure 3, and Table 2, and for creating Pourpre. Thanks to Ozlem Uzuner and Sue Felshin for valuable comments on earlier drafts of this paper and to Boris Katz for his inspiration and support.

[8] http://www-24.nist.gov/projects/aquaint/opinion.html

[9] http://www.darpa.mil/ipto/programs/gale

References

Eric J. Breck, John D. Burger, Lisa Ferro, Lynette Hirschman, David House, Marc Light, and Inderjeet Mani. 2000. How to evaluate your question answering system every day ... and still get real work done. In *Proceedings of the second international conference on Language Res ources and Evaluation (LREC2000)*.

Jimmy Lin and Dina Demner-Fushman. 2005. Automatically evaluating answers to definition questions. In *Proceedings of HLT-EMNLP*.

Jimmy Lin and Dina Demner-Fushman. 2006. Will pyramids built of nuggets topple over? In *Proceedings of HLT-NAACL*.

Jimmy Lin, Eileen Abels, Dina Demner-Fushman, Douglas W. Oard, Philip Wu, and Yejun Wu. 2005. A menagerie of tracks at maryland: HARD, Enterprise, QA, and Genomics, oh my! In *Proceedings of TREC*.

Ellen Voorhees. 2003. Overview of the TREC 2003 question answering track.

Ellen Voorhees. 2004. Overview of the TREC 2004 question answering track.

Ellen Voorhees. 2005. Overview of the TREC 2005 question answering track.

Will Pyramids Built of Nuggets Topple Over?

Jimmy Lin[1,2,3] **and Dina Demner-Fushman**[2,3]
[1]College of Information Studies
[2]Department of Computer Science
[3]Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742, USA
jimmylin@umd.edu, demner@cs.umd.edu

Abstract

The present methodology for evaluating complex questions at TREC analyzes answers in terms of facts called "nuggets". The official F-score metric represents the harmonic mean between recall and precision at the nugget level. There is an implicit assumption that some facts are more important than others, which is implemented in a binary split between "vital" and "okay" nuggets. This distinction holds important implications for the TREC scoring model—essentially, systems only receive credit for retrieving vital nuggets—and is a source of evaluation instability. The upshot is that for many questions in the TREC testsets, the median score across all submitted runs is zero. In this work, we introduce a scoring model based on judgments from multiple assessors that captures a more refined notion of nugget importance. We demonstrate on TREC 2003, 2004, and 2005 data that our "nugget pyramids" address many shortcomings of the present methodology, while introducing only minimal additional overhead on the evaluation flow.

1 Introduction

The field of question answering has been moving away from simple "factoid" questions such as "Who invented the paper clip?" to more complex information needs such as "Who is Aaron Copland?" and "How have South American drug cartels been using banks in Liechtenstein to launder money?", which cannot be answered by simple named-entities. Over the past few years, NIST through the TREC QA tracks has implemented an evaluation methodology based on the notion of "information nuggets" to assess the quality of answers to such complex questions. This paradigm has gained widespread acceptance in the research community, and is currently being applied to evaluate answers to so-called "definition", "relationship", and "opinion" questions.

Since quantitative evaluation is arguably the single biggest driver of advances in language technologies, it is important to closely examine the characteristics of a scoring model to ensure its fairness, reliability, and stability. In this work, we identify a potential source of instability in the nugget evaluation paradigm, develop a new scoring method, and demonstrate that our new model addresses some of the shortcomings of the original method. It is our hope that this more-refined evaluation model can better guide the development of technology for answering complex questions.

This paper is organized as follows: Section 2 provides a brief overview of the nugget evaluation methodology. Section 3 draws attention to the vital/okay nugget distinction and the problems it creates. Section 4 outlines our proposal for building "nugget pyramids", a more-refined model of nugget importance that combines judgments from multiple assessors. Section 5 describes the methodology for evaluating this new model, and Section 6 presents our results. A discussion of related issues appears in Section 7, and the paper concludes with Section 8.

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 383–390,
New York, June 2006.

2 Evaluation of Complex Questions

To date, NIST has conducted three large-scale evaluations of complex questions using a nugget-based evaluation methodology: "definition" questions in TREC 2003, "other" questions in TREC 2004 and TREC 2005, and "relationship" questions in TREC 2005. Since relatively few teams participated in the 2005 evaluation of "relationship" questions, this work focuses on the three years' worth of "definition/other" questions. The nugget-based paradigm has been previously detailed in a number of papers (Voorhees, 2003; Hildebrandt et al., 2004; Lin and Demner-Fushman, 2005a); here, we present only a short summary.

System responses to complex questions consist of an unordered set of passages. To evaluate answers, NIST pools answer strings from all participants, removes their association with the runs that produced them, and presents them to a human assessor. Using these responses and research performed during the original development of the question, the assessor creates an "answer key" comprised of a list of "nuggets"—essentially, facts about the target. According to TREC guidelines, a nugget is defined as a fact for which the assessor could make a binary decision as to whether a response contained that nugget (Voorhees, 2003). As an example, relevant nuggets for the target "AARP" are shown in Table 1. In addition to creating the nuggets, the assessor also manually classifies each as either "vital" or "okay". Vital nuggets represent concepts that must be in a "good" definition; on the other hand, okay nuggets contribute worthwhile information about the target but are not essential. The distinction has important implications, described below.

Once the answer key of vital/okay nuggets is created, the assessor goes back and manually scores each run. For each system response, he or she decides whether or not each nugget is present. The final F-score for an answer is computed in the manner described in Figure 1, and the final score of a system run is the mean of scores across all questions. The per-question F-score is a harmonic mean between nugget precision and nugget recall, where recall is heavily favored (controlled by the β parameter, set to five in 2003 and three in 2004 and 2005). Nugget recall is computed solely on vital nuggets (which means no credit is given for returning okay nuggets), while nugget precision is approximated by a length allowance based on the number of both vital and okay nuggets returned. Early in a pilot study, researchers discovered that it was impossible for assessors to enumerate the total set of nuggets contained in a system response (Voorhees, 2003), which corresponds to the denominator in the precision calculation. Thus, a penalty for verbosity serves as a surrogate for precision.

Note that while a question's answer key only needs to be created once, assessors must manually determine if each nugget is present in a system's response. This human involvement has been identified as a bottleneck in the evaluation process, although we have recently developed an automatic scoring metric called POURPRE that correlates well with human judgments (Lin and Demner-Fushman, 2005a).

vital	30+ million members
okay	Spends heavily on research & education
vital	Largest seniors organization
vital	Largest dues paying organization
vital	Membership eligibility is 50+
okay	Abbreviated name to attract boomers
okay	Most of its work done by volunteers
okay	Receives millions for product endorsements
okay	Receives millions from product endorsements

Table 1: Answer nuggets for the target "AARP".

Let

r	# of *vital* nuggets returned in a response
a	# of *okay* nuggets returned in a response
R	# of *vital* nuggets in the answer key
l	# of non-whitespace characters in the entire answer string

Then

$$\text{recall } (\mathcal{R}) = r/R$$

$$\text{allowance } (\alpha) = 100 \times (r + a)$$

$$\text{precision } (\mathcal{P}) = \begin{cases} 1 & \text{if } l < \alpha \\ 1 - \frac{l-\alpha}{l} & \text{otherwise} \end{cases}$$

Finally, the $F_\beta = \dfrac{(\beta^2 + 1) \times \mathcal{P} \times \mathcal{R}}{\beta^2 \times \mathcal{P} + \mathcal{R}}$

$\beta = 5$ in TREC 2003, $\beta = 3$ in TREC 2004, 2005.

Figure 1: Official definition of F-score.

Testset	# q's	1 vital	2 vital
TREC 2003	50	3	10
TREC 2004	64	2	15
TREC 2005	75	5	16

Table 2: Number of questions with few vital nuggets in the different testsets.

3 What's Vital? What's Okay?

Previously, we have argued that the vital/okay distinction is a source of instability in the nugget-based evaluation methodology, especially given the manner in which F-score is calculated (Hildebrandt et al., 2004; Lin and Demner-Fushman, 2005a). Since only vital nuggets figure into the calculation of nugget recall, there is a large "quantization effect" for system scores on topics that have few vital nuggets. For example, on a question that has only one vital nugget, a system cannot obtain a non-zero score unless that vital nugget is retrieved. In reality, whether or not a system returned a passage containing that single vital nugget is often a matter of luck, which is compounded by assessor judgment errors. Furthermore, there does not appear to be any reliable indicators for predicting the importance of a nugget, which makes the task of developing systems even more challenging.

The polarizing effect of the vital/okay distinction brings into question the stability of TREC evaluations. Table 2 shows statistics about the number of questions that have only one or two vital nuggets. Compared to the size of the testset, these numbers are relatively large. As a concrete example, "F16" is the target for question 71.7 from TREC 2005. The only vital nugget is "First F16s built in 1974". The practical effect of the vital/okay distinction in its current form is the number of questions for which the median system score across all submitted runs is zero: 22 in TREC 2003, 41 in TREC 2004, and 44 in TREC 2005.

An evaluation in which the median score for many questions is zero has many shortcomings. For one, it is difficult to tell if a particular run is "better" than another—even though they may be very different in other salient properties such as length, for example. The discriminative power of the present F-score measure is called into question: are present systems that bad, or is the current scoring model insufficient to discriminate between different (poorly performing) systems?

Also, as pointed out by Voorhees (2005), a score distribution heavily skewed towards zero makes meta-analysis of evaluation stability hard to perform. Since such studies depend on variability in scores, evaluations would appear more stable than they really are.

While there are obviously shortcomings to the current scheme of labeling nuggets as either "vital" or "okay", the distinction does start to capture the intuition that "not all nuggets are created equal". Some nuggets are inherently more important than others, and this should be reflected in the evaluation methodology. The solution, we believe, is to solicit judgments from multiple assessors and develop a more refined sense of nugget importance. However, given finite resources, it is important to balance the amount of additional manual effort required with the gains derived from those efforts. We present the idea of building "nugget pyramids", which addresses the shortcomings noted here, and then assess the implications of this new scoring model against data from TREC 2003, 2004, and 2005.

4 Building Nugget Pyramids

As previously pointed out (Lin and Demner-Fushman, 2005b), the question answering and summarization communities are converging on the task of addressing complex information needs from complementary perspectives; see, for example, the recent DUC task of query-focused multi-document summarization (Amigó et al., 2004; Dang, 2005). From an evaluation point of view, this provides opportunities for cross-fertilization and exchange of fresh ideas. As an example of this intellectual discourse, the recently-developed POURPRE metric for automatically evaluating answers to complex questions (Lin and Demner-Fushman, 2005a) employs n-gram overlap to compare system responses to reference output, an idea originally implemented in the ROUGE metric for summarization evaluation (Lin and Hovy, 2003). Drawing additional inspiration from research on summarization evaluation, we adapt the pyramid evaluation scheme (Nenkova and Passonneau, 2004) to address the shortcomings of

the vital/okay distinction in the nugget-based evaluation methodology.

The basic intuition behind the pyramid scheme (Nenkova and Passonneau, 2004) is simple: the importance of a fact is directly related to the number of people that recognize it as such (i.e., its popularity). The evaluation methodology calls for assessors to annotate Semantic Content Units (SCUs) found within model reference summaries. The weight assigned to an SCU is equal to the number of annotators that have marked the particular unit. These SCUs can be arranged in a pyramid, with the highest-scoring elements at the top: a "good" summary should contain SCUs from a higher tier in the pyramid before a lower tier, since such elements are deemed "more vital".

This pyramid scheme can be easily adapted for question answering evaluation since a nugget is roughly comparable to a Semantic Content Unit. We propose to build nugget pyramids for answers to complex questions by soliciting vital/okay judgments from multiple assessors, i.e., take the original reference nuggets and ask different humans to classify each as either "vital" or "okay". The weight assigned to each nugget is simply equal to the number of different assessors that deemed it vital. We then normalize the nugget weights (per-question) so that the maximum possible weight is one (by dividing each nugget weight by the maximum weight of that particular question). Therefore, a nugget assigned "vital" by the most assessors (not necessarily all) would receive a weight of one.[1]

The introduction of a more granular notion of nugget importance should be reflected in the calculation of F-score. We propose that nugget recall be modified to take into account nugget weight:

$$\mathcal{R} = \frac{\sum_{m \in A} w_m}{\sum_{n \in V} w_n}$$

Where A is the set of reference nuggets that are matched within a system's response and V is the set of all reference nuggets; w_m and w_n are the weights of nuggets m and n, respectively. Instead of a binary distinction based solely on matching vital nuggets, all nuggets now factor into the calculation of recall, subjected to a weight. 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.

We propose to leave the calculation of nugget precision as is: a system would receive a length allowance of 100 non-whitespace characters for every nugget it retrieved (regardless of importance). Longer answers would be penalized for verbosity.

Having outlined our revisions to the standard nugget-based scoring method, we will proceed to describe our methodology for evaluating this new model and demonstrate how it overcomes many of the shortcomings of the existing paradigm.

5 Evaluation Methodology

We evaluate our methodology for building "nugget pyramids" using runs submitted to the TREC 2003, 2004, and 2005 question answering tracks (2003 "definition" questions, 2004 and 2005 "other" questions). There were 50 questions in the 2003 testset, 64 in 2004, and 75 in 2005. In total, there were 54 runs submitted to TREC 2003, 63 to TREC 2004, and 72 to TREC 2005. NIST assessors have manually annotated nuggets found in a given system's response, and this allows us to calculate the final F-score under different scoring models.

We recruited a total of nine different assessors for this study. Assessors consisted of graduate students in library and information science and computer science at the University of Maryland as well as volunteers from the question answering community (obtained via a posting to NIST's TREC QA mailing list). Each assessor was given the reference nuggets along with the original questions and asked to classify each nugget as vital or okay. They were purposely asked to make these judgments without reference to documents in the corpus in order to expedite the assessment process—our goal is to propose a refinement to the current nugget evaluation methodology that addresses shortcomings while minimizing the amount of additional effort required. Combined with the answer key created by the original NIST assessors, we obtained a total of ten judgments for every single nugget in the three testsets.[2]

[1] Since there may be multiple nuggets with the highest score, what we're building is actually a frustum sometimes. :)

[2] Raw data can be downloaded at the following URL: http://www.umiacs.umd.edu/∼jimmylin

	2003		2004		2005	
Assessor	Kendall's τ	zeros	Kendall's τ	zeros	Kendall's τ	zeros
0	1.00	22	1.00	41	1.00	44
1	0.908	20	0.933	36	0.888	43
2	0.896	21	0.916	43	0.900	41
3	0.903	21	0.917	38	0.897	39
4	0.912	20	0.914	42	0.879	56
5	0.873	23	0.926	40	0.841	53
6	0.889	29	0.908	32	0.894	39
7	0.900	22	0.930	37	0.890	54
8	0.909	18	0.932	29	0.891	35
9	0.879	26	0.908	49	0.877	58
average	0.896	22.2	0.920	38.7	0.884	46.2

Table 3: Kendall's τ correlation between system scores generated using "official" vital/okay judgments and each assessor's judgments. (Assessor 0 represents the original NIST assessors.)

We measured the correlation between system ranks generated by different scoring models using Kendall's τ, a commonly-used rank correlation measure in information retrieval for quantifying the similarity between different scoring methods. Kendall's τ computes the "distance" between two rankings as the minimum number of pairwise adjacent swaps necessary to convert one ranking into the other. This value is normalized by the number of items being ranked such that two identical rankings produce a correlation of 1.0; the correlation between a ranking and its perfect inverse is -1.0; and the expected correlation of two rankings chosen at random is 0.0. Typically, a value of greater than 0.8 is considered "good", although 0.9 represents a threshold researchers generally aim for.

We hypothesized that system ranks are relatively unstable with respect to individual assessor's judgments. That is, how well a given system scores is to a large extent dependent on which assessor's judgments one uses for evaluation. This stems from an inescapable fact of such evaluations, well known from studies of relevance in the information retrieval literature (Voorhees, 1998). Humans have legitimate differences in opinion regarding a nugget's importance, and there is no such thing as "the correct answer". However, we hypothesized that these variations can be smoothed out by building "nugget pyramids" in the manner we described. Nugget weights reflect the combined judgments of many individual assessors, and scores generated with weights taken into account should correlate better with each individual assessor's opinion.

6 Results

To verify our hypothesis about the instability of using any individual assessor's judgments, we calculated the Kendall's τ correlation between system scores generated using the "official" vital/okay judgments (provide by NIST assessors) and each individual assessor's judgments. This is shown in Table 3. The original NIST judgments are listed as "assessor 0" (and not included in the averages). For all scoring models discussed in this paper, we set β, the parameter that controls the relative importance of precision and recall, to three.[3] Results show that although official rankings generally correlate well with rankings generated by our nine additional assessors, the agreement is far from perfect. Yet, in reality, the opinions of our nine assessors are not any less valid than those of the NIST assessors—NIST does not occupy a privileged position on what constitutes a good "definition". We can see that variations in human judgments do not appear to be adequately captured by the current scoring model.

Table 3 also shows the number of questions for which systems' median score was zero based on each individual assessor's judgments (out of 50

[3]Note that $\beta = 5$ in the official TREC 2003 evaluation.

	2003	2004	2005
0	0.934	0.943	0.901
1	0.962	0.940	0.950
2	0.938	0.948	0.952
3	0.938	0.947	0.950
4	0.936	0.922	0.914
5	0.916	0.956	0.887
6	0.916	0.950	0.958
7	0.949	0.933	0.927
8	0.964	0.972	0.953
9	0.912	0.899	0.881
average	0.936	0.941	0.927

Table 4: Kendall's τ correlation between system rankings generated using the ten-assessor nugget pyramid and those generated using each individual assessor's judgments. (Assessor 0 represents the original NIST assessors.)

questions for TREC 2003, 64 for TREC 2004, and 75 for TREC 2005). These numbers are worrisome: in TREC 2004, for example, over half the questions (on average) have a median score of zero, and over three quarters of questions, according to assessor 9. This is problematic for the various reasons discussed in Section 3.

To evaluate scoring models that combine the opinions of multiple assessors, we built "nugget pyramids" using all ten sets of judgments in the manner outlined in Section 4. All runs submitted to each of the TREC evaluations were then rescored using the modified F-score formula, which takes into account a finer-grained notion of nugget importance. Rankings generated by this model were then compared against those generated by each individual assessor's judgments. Results are shown in Table 4. As can be seen, the correlations observed are higher than those in Table 3, meaning that a nugget pyramid better captures the opinions of each individual assessor. A two-tailed t-test reveals that the differences in averages are statistically significant ($p << 0.01$ for TREC 2003/2005, $p < 0.05$ for TREC 2004).

What is the effect of combining judgments from different numbers of assessors? To answer this question, we built ten different nugget pyramids of varying "sizes", i.e., combining judgments from one through ten assessors. The Kendall's τ correlations between scores generated by each of these and scores generated by each individual assessor's judgments were computed. For each pyramid, we computed the average across all rank correlations, which captures the extent to which that particular pyramid represents the opinions of all ten assessors. These results are shown in Figure 2. The increase in Kendall's τ that comes from adding a second assessor is statistically significant, as revealed by a two-tailed t-test ($p << 0.01$ for TREC 2003/2005, $p < 0.05$ for TREC 2004), but ANOVA reveals no statistically significant differences beyond two assessors.

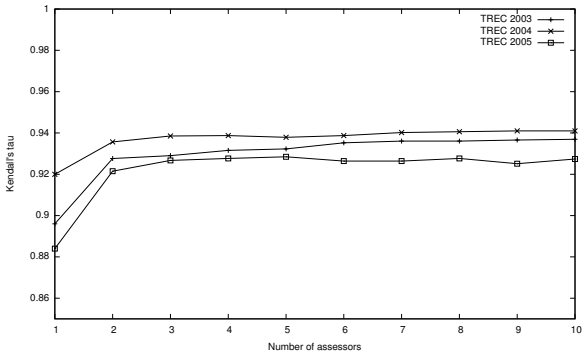


Figure 2: Average agreement (Kendall's τ) between individual assessors and nugget pyramids built from different numbers of assessors.

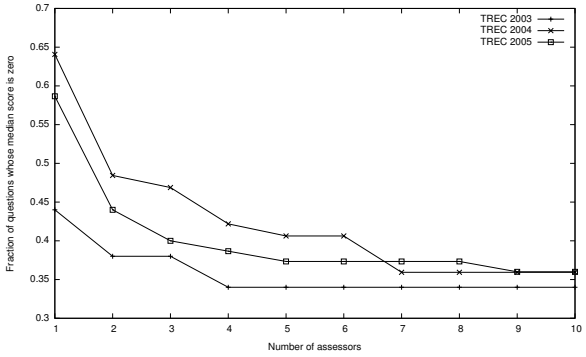


Figure 3: Fraction of questions whose median score is zero plotted against number of assessors whose judgments contributed to the nugget pyramid.

From these results, we can conclude that adding a second assessor yields a scoring model that is significantly better at capturing the variance in human relevance judgments. In this respect, little is gained beyond two assessors. If this is the only advantage

provided by nugget pyramids, then the boost in rank correlations may not be sufficient to justify the extra manual effort involved in building them. As we shall see, however, nugget pyramids offer other benefits as well.

Evaluation by our nugget pyramids greatly reduces the number of questions whose median score is zero. As previously discussed, a strict vital/okay split translates into a score of zero for systems that do not return any vital nuggets. However, nugget pyramids reflect a more refined sense of nugget importance, which results in fewer zero scores. Figure 3 shows the number of questions whose median score is zero (normalized as a fraction of the entire testset) by nugget pyramids built from varying numbers of assessors. With four or more assessors, the number of questions whose median is zero for the TREC 2003 testset drops to 17; for TREC 2004, 23 for seven or more assessors; for TREC 2005, 27 for nine or more assessors. In other words, F-scores generated using our methodology are far more discriminative. The remaining questions with zero medians, we believe, accurately reflect the state of the art in question answering performance.

An example of a nugget pyramid that combines the opinions of all ten assessors is shown in Table 5 for the target "AARP". Judgments from the original NIST assessors are also shown (cf. Table 1). Note that there is a strong correlation between the original vital/okay judgments and the refined nugget weights based on the pyramid, indicating that (in this case, at least) the intuition of the NIST assessor matches that of the other assessors.

1.0	vital	Largest seniors organization
0.9	vital	Membership eligibility is 50+
0.8	vital	30+ million members
0.7	vital	Largest dues paying organization
0.2	okay	Most of its work done by volunteers
0.1	okay	Spends heavily on research & education
0.1	okay	Receives millions for product endorsements
0.1	okay	Receives millions from product endorsements
0.0	okay	Abbreviated name to attract boomers

Table 5: Answer nuggets for the target "AARP" with weights derived from the nugget pyramid building process.

7 Discussion

In balancing the tradeoff between advantages provided by nugget pyramids and the additional manual effort necessary to create them, what is the optimal number of assessors to solicit judgments from? Results shown in Figures 2 and 3 provide some answers. In terms of better capturing different assessors' opinions, little appears to be gained from going beyond two assessors. However, adding more judgments does decrease the number of questions whose median score is zero, resulting in a more discriminative metric. Beyond five assessors, the number of questions with a zero median score remains relatively stable. We believe that around five assessors yield the smallest nugget pyramid that confers the advantages of the methodology.

The idea of building "nugget pyramids" is an extension of a similarly-named evaluation scheme in document summarization, although there are important differences. Nenkova and Passonneau (2004) call for multiple assessors to annotate SCUs, which is much more involved than the methodology presented here, where the nuggets are fixed and assessors only provide additional judgments about their importance. This obviously has the advantage of streamlining the assessment process, but has the potential to miss other important nuggets that were not identified in the first place. Our experimental results, however, suggest that this is a worthwhile tradeoff. The explicit goal of this work was to develop scoring models for nugget-based evaluation that would address shortcomings of the present approach, while introducing minimal overhead in terms of additional resource requirements. To this end, we have been successful.

Nevertheless, there are a number of issues that are worth mentioning. To speed up the assessment process, assessors were instructed to provide "snap judgments" given only the list of nuggets and the target. No additional context was provided, e.g., documents from the corpus or sample system responses. It is also important to note that the reference nuggets were never meant to be read by other people—NIST makes no claim for them to be well-formed descriptions of the facts themselves. These answer

keys were primarily note-taking devices to assist in the assessment process. The important question, however, is whether scoring variations caused by poorly-phrased nuggets are smaller than the variations caused by legitimate inter-assessor disagreement regarding nugget importance. Our experiments appear to suggest that, overall, the nugget pyramid scheme is sound and can adequately cope with these difficulties.

8 Conclusion

The central importance that quantitative evaluation plays in advancing the state of the art in language technologies warrants close examination of evaluation methodologies themselves to ensure that they are measuring "the right thing". In this work, we have identified a shortcoming in the present nugget-based paradigm for assessing answers to complex questions. The vital/okay distinction was designed to capture the intuition that some nuggets are more important than others, but as we have shown, this comes at a cost in stability and discriminative power of the metric. We proposed a revised model that incorporates judgments from multiple assessors in the form of a "nugget pyramid", and demonstrated how this addresses many of the previous shortcomings. It is hoped that our work paves the way for more accurate and refined evaluations of question answering systems in the future.

9 Acknowledgments

This work has been supported in part by DARPA contract HR0011-06-2-0001 (GALE), and has greatly benefited from discussions with Ellen Voorhees, Hoa Dang, and participants at TREC 2005. We are grateful for the nine assessors who provided nugget judgments. The first author would like to thank Esther and Kiri for their loving support.

References

Enrique Amigó, Julio Gonzalo, Victor Peinado, Anselmo Peñas, and Felisa Verdejo. 2004. An empirical study of information synthesis task. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004)*.

Hoa Dang. 2005. Overview of DUC 2005. In *Proceedings of the 2005 Document Understanding Conference (DUC 2005) at NLT/EMNLP 2005*.

Wesley Hildebrandt, Boris Katz, and Jimmy Lin. 2004. Answering definition questions with multiple knowledge sources. In *Proceedings of the 2004 Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT/NAACL 2004)*.

Jimmy Lin and Dina Demner-Fushman. 2005a. Automatically evaluating answers to definition questions. In *Proceedings of the 2005 Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*.

Jimmy Lin and Dina Demner-Fushman. 2005b. Evaluating summaries and answers: Two sides of the same coin? In *Proceedings of the ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT/NAACL 2003)*.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *Proceedings of the 2004 Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT/NAACL 2004)*.

Ellen M. Voorhees. 1998. Variations in relevance judgments and the measurement of retrieval effectiveness. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1998)*.

Ellen M. Voorhees. 2003. Overview of the TREC 2003 question answering track. In *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003)*.

Ellen M. Voorhees. 2005. Using question series to evaluate question answering system effectiveness. In *Proceedings of the 2005 Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*.

Creating a Test Collection for Citation-based IR Experiments

Anna Ritchie
University of Cambridge
Computer Laboratory
15 J J Thompson Avenue
Cambridge, CB3 0FD, U.K.
ar283@cl.cam.ac.uk

Simone Teufel
University of Cambridge
Computer Laboratory
15 J J Thompson Avenue
Cambridge, CB3 0FD, U.K.
sht25@cl.cam.ac.uk

Stephen Robertson
Microsoft Research Ltd
Roger Needham House
7 J J Thomson Avenue
Cambridge, CB3 0FB, U.K.
ser@microsoft.com

Abstract

We present an approach to building a test collection of research papers. The approach is based on the Cranfield 2 tests but uses as its vehicle a current conference; research questions and relevance judgements of all cited papers are elicited from conference authors. The resultant test collection is different from TREC's in that it comprises scientific articles rather than newspaper text and, thus, allows for IR experiments that include citation information. The test collection currently consists of 170 queries with relevance judgements; the document collection is the ACL Anthology. We describe properties of our queries and relevance judgements, and demonstrate the use of the test collection in an experimental setup. One potentially problematic property of our collection is that queries have a low number of relevant documents; we discuss ways of alleviating this.

1 Introduction

We present a methodology for creating a test collection of scientific papers that is based on the Cranfield 2 methodology but uses a current conference as the main vehicle for eliciting relevance judgements from users, i.e., the authors.

Building a test collection is a long and expensive process but was necessary as no ready-made test collection existed on which the kinds of experiments with citation information that we envisage could be run. We aim to improve term-based IR on scientific articles with citation information, by using index terms from the citing article to additionally describe the cited document. Exactly how to do this is the research question that our test collection should help to address.

This paper is structured as follows: Section 2 motivates our proposed experiments and, thereby, our test collection. Section 3 discusses the how test collections are built and, in particular, our own. Section 4 briefly describes the practicalities of compiling the document collection and the processing we perform to prepare the documents for our experiments. In Section 5, we show that our test collection can be used with standard IR tools. Finally, Section 6 discusses the problem of the low number of relevant documents judged so far and two ways of alleviating this problem.

2 Motivation

The idea of using terms external to a document, coming from a 'citing' document, has been borrowed from web-based IR. When one paper cites another, a link is made between them and this *link structure* is analogous to that of the web: "hyperlinks ... provide semantic linkages between objects, much in the same manner that citations link documents to other related documents" (Pitkow and Pirolli, 1997). Link structure, particularly anchor text, has been used to advantage in web-based IR. While web pages are often poorly self-descriptive (Brin and Page, 1998) anchor text is often a higher-level description of the pointed-to page. (Davison,

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 391–398,
New York, June 2006.

2000) provides a good discussion of how well anchor text does this and provides experimental results in support. Thus, beginning with (McBryan, 1994), there is a trend of propagating anchor text along its hyperlink to associate it with the linked page, as well as the page in which it is found. Google, for example, includes anchor text as index terms for the linked page (Brin and Page, 1998). The TREC Web tracks have also shown that using anchor text improves retrieval effectiveness for some search tasks (Hawking and Craswell, 2005).

This idea has already been applied to citations and scientific articles (Bradshaw, 2003). In Bradshaw's experiment, scientific documents are indexed by the text that refers to them in documents that cite them. However, unlike in experiments with previous collections, we need both the citing and the cited article as full documents in our collection. The question of how to identify citation 'anchor text' and its extent is a matter for research; this requires the full text of the citing article. Previous experiments and test collections have had only limited access to the content of the citing article: Bradshaw had access only to a fixed window of text around the citation, as provided by CiteSeer's 'citation context'; in the GIRT collections (Kluck, 2003), a dozen or so content-bearing information fields (e.g., title, abstract, methodological descriptors) represent each document and the full text is not available. Additionally, in Bradshaw's experiment, no access is given to the text of the *cited* article itself so that the influence of a term-based IR model cannot be studied and so that documents can only be indexed if they have been cited at least once. A test collection containing full text for many citing and cited documents, thus, has advantages from a methodological point of view.

2.1 Choosing a Genre

When choosing a scientific field to study, we looked for one that is practicable for us to compile the document collection (freely available machine-readable documents; as few as possible document styles), while still ensuring good coverage of research topics in an entire field. Had we chosen the medical field or bioinformatics, the prolific number of journals would have been a problem for the practical document preparation.

We also looked for a relatively self-contained field. As we aim to propagate referential text to cited papers as index terms, references from documents in the collection to other documents *within* the collection will be most useful. We call these *internal* references. While it is impossible to find or create a collection of documents with only internal references, we aim for as high a proportion of internal references as possible.

We chose the ACL (Association for Computational Linguistics) Anthology[1], a freely available digital archive of computational linguistics research papers. Computational linguistics is a small, homogenous research field and the Anthology contains the most prominent publications since the beginning of the field in 1960, consists of only 2 journals, 7 conferences and 5 less important publications, such as discontinued conferences and a series of workshops, resulting in only 7000 papers[2].

With the ACL Anthology, we expect a high proportion of internal references within a relatively compact document collection. We empirically measured the proportion of collection-internal references. We found a proportion of internal references to all references of 0.33 (the *in-factor*). We wanted to compare this number to a situation in another, larger field (genetics) but no straightforward comparison is possible, as there are very many genetics journals and quality of journals probably plays a larger role in a bigger field. We tried to simulate a similar collection to the 9 main journals+conferences in the Anthology, by considering 10 journals in genetics with a range of impact factors[3], resulting in an in-factor of 0.17 (dropping to 0.14 if only 5 journals are considered). Thus, our hypothesis that the Anthology is reasonably self-contained, at least in comparison with other possible collections, was confirmed.

The choice of computational linguistics has the added benefit that we are familiar with the domain; we can interpret the subject matter better than we would be able to in the medical domain. This should be of use to us in our eventual experiments.

[1] http://www.aclweb.org/anthology/

[2] This is our estimate, after substracting non-papers such as letters to the editor, tables of contents etc. The Anthology is growing by $\sim$500 papers per year.

[3] Journal impact factor is a measure of the frequency with which its average article is cited and is a measure of the relative importance of journals within a field (Garfield, 1972).

3 Building Test Collections

To turn our document collection into a test collection, a parallel set of search queries and relevance judgements is needed. There are a number of alternative methods for building a test collection. For TREC, humans devise queries specifically for a given set of documents and make relevance judgements on pooled retrieved documents from that set (Harman, 2005). Theirs is an extremely labour-intensive and expensive process and an unrealistic option in the context of our project.

The Cranfield 2 tests (Cleverdon et al., 1966) introduced an alternative method for creating a test collection, specifically for scientific texts. The method was subject to criticism and has not been employed much since. Nevertheless, we believe this method to be worth revisiting for our current situation. In this section, we describe in turn the Cranfield 2 method and our adapted method. We discuss some of the original criticisms and their bearing on our own work, then describe our returns thus far.

3.1 The Cranfield 2 Test Collection

The Cranfield 2 tests (Cleverdon et al., 1966) were a comparative evaluation of indexing language devices. From a base collection of 182 (high speed aerodynamics and aircraft structures) papers, the Cranfield test collection was built by asking the authors to formulate the research question(s) behind their work and to judge how relevant each reference in their paper was to each of their research questions, on a 5-point scale. Referenced documents were obtained and added to the base set. Authors were also asked to list additional relevant papers not cited in their paper. The collection was further expanded in a second stage, using bibliographic coupling to search for similar papers to the referenced ones and employing humans to search the collection for other relevant papers. The resultant collection comprised 1400 documents and 221 queries (Cleverdon, 1997).

The principles behind the Cranfield technique are:

- Queries: Each paper has an underlying research question or questions; these constitute valid search queries.
- Relevant documents: A paper's reference list is a good starting point for finding papers relevant to its research questions.
- Judges: The paper author is the person best qualified to judge relevance.

3.2 Our Anthology Test Collection

We altered the Cranfield design to fit to a fixed, existing document collection. We designed our methodology around an upcoming conference and approached the paper authors at around the time of the conference, to maximize their willingness to participate and to minimise possible changes in their perception of relevance since they wrote the paper. Due to the relatively high in-factor of the collection, we expected a significant proportion of the relevance judgements gathered in this way to be about Anthology documents and, thus, useful as evaluation data.

Hence, the authors of accepted papers for ACL-2005 and HLT-EMNLP-2005 were asked, by email, for their research questions and relevance judgements for their references. We defined a 4-point relevance scale, c.f. Table 1, since we felt that the distinctions between the Cranfield grades were not clear enough to warrant 5. Our guidelines also included examples of referencing situations that might fit each category. Personalized materials for participation were sent, including a reproduction of their paper's reference list in their response form. This meant that invitations could only be sent once the paper had been made available online.

We further deviated from the Cranfield methodology by deciding not to ask the authors to try to list additional references that could have been included in their reference list. An author's willingness to name such references will differ more from author to author than their naming of original references, as referencing is part of a standardized writing process. By asking for this data, the consistency of the data across papers will be degraded and the status of any additional references will be unclear. Furthermore, feedback from an informal pilot study conducted on ten paper authors confirmed that some authors found this task particularly difficult.

Each co-author of the papers was invited individually to participate, rather than inviting the first author alone. This increased the number of invitations that needed to be prepared and sent (by a factor of around 2.5) but also increased the likelihood of getting a return for a given paper. Furthermore, data from multiple co-authors of the same paper can be used to

Grade	Description
4	The reference is crucially relevant to the problem. Knowledge of the contents of the referred work will be fundamental to the reader's understanding of your paper. Often, such relevant references are afforded a substantial amount of text in a paper e.g., a thorough summary.
3	The reference is relevant to the problem. It may be helpful for the reader to know the contents of the referred work, but not crucial. The reference could not have been substituted or dropped without making significant additions to the text. A few sentences may be associated with the reference.
2	The reference is somewhat (perhaps indirectly) relevant to the problem. Following up the reference probably would not improve the reader's understanding of your paper. Alternative references may have been equally appropriate (e.g., the reference was chosen as a representative example from a number of similar references or included in a list of similar references). Or the reference could have been dropped without damaging the informativeness of your paper. Minimal text will be associated with the reference.
1	The reference is irrelevant to this particular problem.

Table 1: Relevance Scale

measure co-author agreement on the relevance task. This is an interesting research question, as it is not at all clear how much even close collaborators would agree on relevance, but we do not address this here.

We plan to expand the collection in a second stage, in line with the Cranfield 2 design. We will reapproach contributing authors after obtaining retrieval results on our collection (e.g., with a standard IR engine) and ask them to make additional relevance judgements on these papers.

3.3 Criticisms of Cranfield 2

Both Cranfield 1 (Cleverdon, 1960) and 2 were subject to various criticisms; (Spärck Jones, 1981) gives an excellent account of the tests and their criticisms. The majority were criticisms of the test collection paradigm itself and are not pertinent here. However, the *source-document principle* (i.e., the use of queries created from documents in the collection) attracted particular criticisms. The fundamental concern was that the way in which the queries were created led to "an unnaturally close relation" between the terms in the queries and those used to index the documents in the colection (Vickery, 1967); any such relationship might have created a bias towards a particular indexing language, distorting the comparisons that were the goal of the project.

In Cranfield 1, system success was measured by retrieval of source documents alone, criticized for being an over-simplification and a distortion of 'real-life' searching. The evaluation procedure was changed for Cranfield 2 so that source documents were excluded from searches and, instead, retrieval of other relevant documents was used to measure success. This removed the problem that, usually, when a user searches, there is no source document for their query. Despite this, Vickery notes that there were "still verbal links between sought document and question" in the new method: each query author was asked to judge the relevance of the source document's references and "the questions ... were formulated *after* the cited papers had been read and has possibly influenced the wording of his question".

While adapting the Cranfield 2 method to our needs, we have tried to address some of the criticisms, e.g., that authors' relevance judgements change over time. Nevertheless, we still have source-document queries and must consider the associated criticisms. Firstly, our test collection is not intended for comparisons of indexing languages. Rather, we aim to compare the effect of adding extra index terms to a base indexing of the documents. The source documents will have no influence on the base indexing of a document above that of the other documents. The additional index terms, coming from citations to that document, will generally be 'chosen' by someone other than the query author, with no knowledge of the query terms[4]. Also, our documents will be indexed fully automatically, further diminishing the scope of any subconscious human influence.

Thus, we believe that the suspect relationship between queries and indexing is negligible in the con-

[4]The exception to this is self-citation. This (very indirectly) allows the query author to influence the indexing but it seems highly improbable that an author would be thinking about their query whilst citing a previous work.

text of our work, as opposed to the Cranfield tests, and that the source-document principle is sound.

3.4 Returns and Analysis

Out of around 500 invitations sent to conference authors, 85 resulted in research questions with relevance judgements being returned; 235 queries in total. Example queries are:

- *Do standard probabilistic parsing techniques, developed for English, fare well for French and does lexicalistion help improve parsing results?*
- *Analyze the lexical differences between genders engaging in telephone conversations.*

Of the 235 queries, 18 were from authors whose co-authors had also returned data and were discarded (for retrieval purposes); we treat co-author data on the same paper as 'the same' and keep only the first authors'. 47 queries had no relevant Anthology-internal references and were discarded. Another 15 had only relevant Anthology references not yet included in the archive[5]; we keep these for the time being. This leaves 170 unique queries with at least 1 relevant Anthology reference and an average of 3.8 relevant Anthology references each. The average in-factor across queries is 0.42 (similar to our previously estimated Anthology in-factor)[6].

Our average number of judged relevant documents per query is lower than for Cranfield, which had an average of 7.2 (Spärck Jones et al., 2000). However, this is the final number for the Cranfield collection, arrived at after the second stage of relevance judging, which we have not yet carried out. Nevertheless, we must anticipate a potentially low number of relevant documents per query, particularly in comparison to, e.g., the TREC ad hoc track (Voorhees and Harman, 1999), with 86.8 judged relevant documents per query.

4 Document Collection and Processing

The Anthology documents are distributed in PDF, a format designed to visually render printable documents, not to preserve editable text. So the PDF collection must be converted into a fully textual format.

[5]HLT-NAACL-2004 papers, e.g., are listed as 'in process'.

[6]We cannot directly compare this to Cranfield's in-factor as we do not have access to the documents.

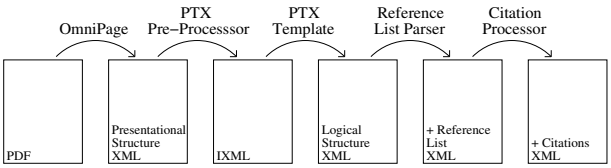


Figure 1: Document Processing Pipeline

A pipeline of processing stages has been developed in the framework of a wider project, illustrated in Figure 1.

Firstly, OmniPage Pro 14[7], a commercial PDF processing software package, scans the PDFs and produces an XML encoding of character-level page layout information. AI algorithms for heuristically extracting character information (similar to OCR) are necessary since many of the PDFs were created from scanned paper-copies and others do not contain character information in an accessible format.

The OmniPage output describes a paper as text blocks with typesetting information such as font and positional information. A pre-processor (Lewin et al., 2005) filters and summarizes the OmniPage output into Intermediate XML (IXML), as well as correcting certain characteristic errors from that stage. A journal-specific template converts the IXML to a logical XML-based document structure (Teufel and Elhadad, 2002), by exploiting low-level, presentational, journal-specific information such as font size and positioning of text blocks.

Subsequent stages incrementally add more detailed information to the logical representation. The paper's reference list is annotated in more detail, marking up individual references, author names, titles and years of publication. Finally, a citation processor identifies and marks up citations in the document body and their constituent parts, e.g., author names and years.

5 Preliminary Experimentation

We expect that our test collection, built for our citation experiments, will be of wider value and we intend to make it publicly available. As a sanity check on our data so far, we carried out some preliminary experimentation, using standard IR tools: the Lemur Toolkit[8], specifically Indri (Strohman et al., 2005),

[7]http://www.scansoft.com/omnipage/

[8]http://www.lemurproject.org/

its integrated language-model based search engine, and the TREC evaluation software, trec_eval[9].

5.1 Experimental Set-up

We indexed around 4200 Anthology documents. This is the total number of documents that have, at the time of writing, been processed by our pipeline (24 years of CL journal, 25 years of ACL proceedings, 14 years of assorted workshops), plus another ∼90 documents for which we have relevance judgements that are not currently available through the Anthology website but should be incorporated into the archive in the future. The indexed documents do not yet contain annotation of the reference list or citations in text. 19 of our 170 queries have no relevant references in the indexed documents and were not included in these experiments. Thus, Figure 2 shows the distribution of queries over number of relevant Anthology references, for a total of 151 queries.

Our Indri index was built using default parameters with no optional processing, e.g., stopping or stemming, resulting in a total of 20117410 terms, 218977 unique terms and 2263 'frequent'[10] terms.

We then prepared an Indri-style query file from the conference research questions. The Indri query language is designed to handle highly complex queries but, for our very basic purposes, we created simple bag-of-words queries by stripping all punctuation from the natural language questions and using Indri's `#combine` operator over all the terms. This means Indri ranks documents in accordance with query likelihood. Again, no stopping or stemming was applied.

Next, the query file was run against the Anthology index using IndriRunQuery with default parameters and, thus, retrieving 1000 documents for each query.

Finally, for evaluation, we converted the Indri's ranked document lists to TREC-style top_results file and the conference relevance judgements compiled into a TREC-style qrels file, including only judgements corresponding to references within the indexed documents. These files were then input to trec_eval, to calculate precision and recall metrics.

[9]http://trec.nist.gov/trec_eval/trec_eval.8.0.tar.gz

[10]Terms that occur in over 1000 documents.

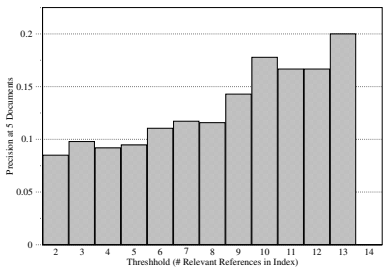


Figure 3: Effect of Thresholding on P at 5 Docs

5.2 Results and Discussion

Out of 489 relevant documents, 329 were retrieved within 1000 (per query) documents. The mean average precision (MAP) was 0.1014 over the 151 queries. This is the precision calculated at each relevant document retrieved (0.0, if that document is not retrieved), averaged over all relevant documents for all queries, i.e., non-interpolated. R-precision, the precision after R (the number of relevant documents for a query) documents are returned, was 0.0965. The average precision at 5 documents was 0.0728.

We investigated the effect of excluding queries with lower than a threshold number of judged relevant documents. Figure 3 shows that precision at 5 documents increases as greater threshold values are applied. Similar trends were observed with other evaluation measures, e.g., MAP and R-precision increased to 0.2018 and 0.1528, respectively, when only queries with 13 or more relevant documents were run, though such stringent thresholding does result in very few queries. Nevertheless, these trends do suggest that the present low number of relevant documents has an adverse effect on retrieval results and is a potential problem for our test collection.

We also investigated the effect of including only authors' main queries, as another potential way of objectively constructing a 'higher quality' query set. Although, this decreased the average in-factor of relevant references, it did, in fact, increase the average absolute number of relevant references in the index. Thus, MAP increased to 0.1165, precision at 5 documents to 0.1016 and R-precision to 0.1201.

These numbers look poor in comparison to the performance of IR systems at TREC but, importantly, they are not intended as performance results. Their purpose is to demonstrate that such numbers *can* be produced using the data we have collected,

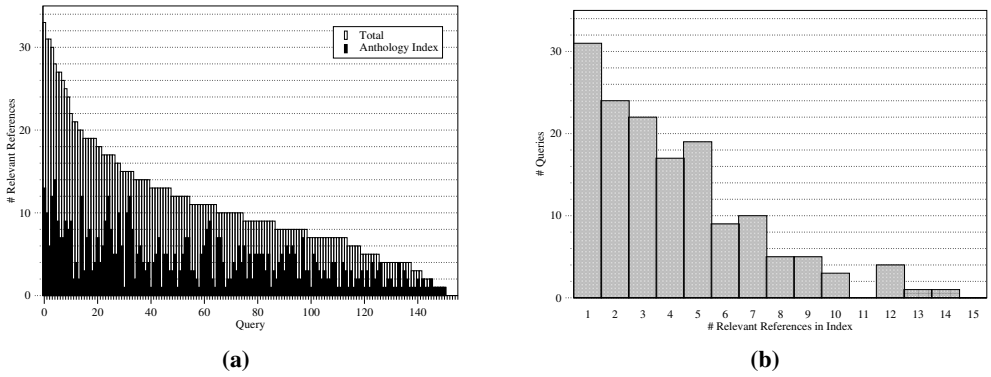


Figure 2: **(a)** Relevant References Per Query and **(b)** Distribution of Queries over Number of Relevant References

rather than to evaluate the performance of some new retrieval system or strategy.

A second point for consideration follows directly from the first: our experiments were carried out on a new test collection and "different test collections have different intrinsic difficulty" (Buckley and Voorhees, 2004). Thus, it is meaningless to compare statistics from this data (from a different domain) to those from the TREC collections, where queries and relevance judgements were collected in a different way, and where there are very many relevant documents.

Thirdly, our experiments used only the most basic techniques and the results could undoubtedly be improved by, e.g., applying a simple stop-list. Nevertheless, this notion of intrinsic difficulty means that it may be the case that evaluations carried out on this collection will produce characteristically low precision values.

Low numbers do not necessarily preclude our data's usefulness as a test collection, whose purpose is to facilitate comparative evaluations. (Voorhees, 1998) states that "To be viable as a laboratory tool, a [test] collection must reliably rank different retrieval variants according to their true effectiveness" and defends the Cranfield paradigm (from criticisms based on relevance subjectivity) by demonstrating that the relative performance of retrieval runs is stable despite differences in relevance judgements. The underlying principle is that it is not the absolute precision values that matter but the ability to compare these values for different retrieval techniques or systems, to investigate their relative benefits. A test collection with low precision values will still allow this.

It is known that all evaluation measures are unstable for very small numbers of relevant documents (Buckley and Voorhees, 2000) and there are issues arising from incomplete relevance information in a test collection (Buckley and Voorhees, 2004). This makes the second stage of our test collection compilation even more indispensable (asking subjects to judge retrieved documents), as this will increase the number of judged relevant documents, as well as bridging the completeness gap.

There are further possibilities of how the problem could be countered. We could exclude queries with lower than a threshold number of relevant documents (after the second stage). Given the respectable number of queries we have, we might be able to afford this luxury. We could add relevant documents from outside the Anthology to our collection. This is least preferable methodologically: using the Anthology has the advantage that it has a real identity and was created for real reasons outside our experiments. Furthermore, the collection 'covers a field', i.e., it includes all important publications and only those. By adding external documents to the collection, it would lose both these properties.

6 Conclusions and Future Work

We have presented an approach to building a test collection from an existing collection of research papers and described the application of our method to the ACL Anthology. We have collected 170 queries with relevance data, centered around the ACL-2005 and HLT-EMNLP-2005 conferences. We

have sanity-checked the usability of our data by running the queries through a retrieval system and evaluating the results using standard software. The collection currently has a low number of judged relevant documents and further experimentation is needed to determine if this poses a real problem.

We plan a second stage of collecting relevance judgements, in line with the original Cranfield design, whereby authors who have contributed queries will be asked to judge the relevance of documents in retrieval rankings from standard IR models and, ideally, from our eventual citation-based experiments.

Nevertheless, our test collection is likely to suffer from incomplete relevance information. The bpref measure (Buckley and Voorhees, 2004) gauges retrieval effectiveness solely on the basis of judged documents and is more stable to differing levels of completeness than measures such as MAP, R-precision or precision at fixed document cutoffs. Thus, bpref may offer a solution to the incompleteness problem and we intend to investigate its potential use in our future evaluations.

When finished, we hope our test collection will be a generally useful IR resource. In particular, we expect the collection to be useful for experimentation with citation information, for which there is currently no existing test collection with the properties that ours offers.

Acknowledgements Thanks to the reviewers for their useful comments and to Karen Spärck Jones for many instructive discussions.

References

Shannon Bradshaw. 2003. Reference directed indexing: Redeeming relevance for subject search in citation indexes. In *Research and Advanced Technology for Digital Libraries (ECDL)*, pages 499–510.

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.

Chris Buckley and Ellen Voorhees. 2000. Evaluating evaluation measure stability. In *Research and Development in Information Retrieval (SIGIR)*.

Chris Buckley and Ellen Voorhees. 2004. Retrieval evaluation with incomplete information. In *Research and development in information retrieval (SIGIR)*.

Cyril Cleverdon, Jack Mills, and Michael Keen. 1966. Factors determining the performance of indexing sytems, volume 1. design. Technical report, ASLIB Cranfield Project.

Cyril Cleverdon. 1960. Report on the first stage of an investigation into the comparative efficiency of indexing systems. Technical report, ASLIB Cranfield Project.

Cyril Cleverdon. 1997. The Cranfield tests on index language devices. In *Readings in information retrieval*, pages 47–59. Morgan Kaufmann Publishers Inc.

Brian D. Davison. 2000. Topical locality in the web. In *Research and Development in Information Retrieval (SIGIR)*, pages 272–279.

Eugene Garfield. 1972. Citation analysis as a tool in journal evaluation. *Science*, 178 (4060):471–479.

Donna Harman. 2005. The TREC test collections. In Ellen Voorhees and Donna Harman, editors, *TREC Experiment and Evaluation in Information Retrieval*, chapter 2. MIT Press.

David Hawking and Nick Craswell. 2005. The very large collection and web tracks. In Ellen Voorhees and Donna Harman, editors, *TREC: Experiment and Evaluation in Information Retrieval*, chapter 9. MIT Press.

Michael Kluck. 2003. The GIRT data in the evaluation of CLIR systems - from 1997 until 2003. In *CLEF*, pages 376–390.

Ian Lewin, Bill Hollingsworth, and Dan Tidhar. 2005. Retrieving hierarchical text structure from typeset scientific articles - a prerequisite for e-science text mining. In *UK e-Science All Hands Meeting*.

Oliver McBryan. 1994. GENVL and WWWW: Tools for taming the web. In *World Wide Web Conference*.

James Pitkow and Peter Pirolli. 1997. Life, death, and lawfulness on the electronic frontier. In *Human Factors in Computing Systems*.

Karen Spärck Jones, Steve Walker, and Stephen Robertson. 2000. A probabilistic model of information retrieval: development and comparative experiments - parts 1 and 2. *Information Processing and Management*, 36(6):779–840.

Karen Spärck Jones. 1981. The Cranfield tests. In Karen Spärck Jones, editor, *Information Retrieval Experiment*, chapter 13, pages 256–284. Butterworths.

Trevor Strohman, Donald Metzler, Howard Turtle, and W. Bruce Croft. 2005. Indri: a language-model based search engine for complex queries. Technical report, University of Massachusetts.

Simone Teufel and Noemie Elhadad. 2002. Collection and linguistic processing of a large-scale corpus of medical articles. In *Language Resources and Evaluation Conference (LREC)*.

B. C. Vickery. 1967. Reviews of CLEVERDON, C. W., MILLS, J. and KEEN, E. M. the Cranfield 2 report. *Journal of Documentation*, 22:247–249.

Ellen Voorhees and Donna Harman. 1999. Overview of the eighth Text REtrieval Conference (TREC 8). In *Text REtrieval Conference (TREC)*.

Ellen Voorhees. 1998. Variations in relevance judgments and the measurement of retrieval effectiveness. In *Research and Development in Information Retrieval (SIGIR)*, pages 315–323.

A Machine Learning based Approach to Evaluating Retrieval Systems

Huyen-Trang Vu and **Patrick Gallinari**
Laboratory of Computer Science (LIP6)
University of Pierre and Marie Curie
8 rue du capitaine Scott, 75015 Paris, France
{vu,gallinar}@poleia.lip6.fr

Abstract

Test collections are essential to evaluate Information Retrieval (IR) systems. The relevance assessment set has been recognized as the key bottleneck in test collection building, especially on very large sized document collections. This paper addresses the problem of efficiently selecting documents to be included in the assessment set. We will show how machine learning techniques can fit this task. This leads to smaller pools than traditional round robin pooling, thus reduces significantly the manual assessment workload. Experimental results on TREC collections[1] consistently demonstrate the effectiveness of our approach according to different evaluation criteria.

1 Introduction

The effectiveness of retrieval systems is often justified by benchmark test collections. A standard test collection consists of lots of documents, a set of information needs, called topics and human judgment about the relevance status of each document for a topic. Nowadays, it is relatively easy to gather huge set of millions of documents and hundreds of topics. The key obstacle for forming large sized test collections lies therefore in the topic assessment procedure. Assessing the whole document sets is unfeasible, even for small sized collection of 800,000 documents (Voorhees and Harman, 1999). In order to keep the assessment process practical, one often selects a certain number of documents for judgment. This is called (document) pooling and the outcome the pool or the qrels (*q*uery *r*elevance *s*et). The collected documents are then judged by humans, documents outside the pool are assumed non relevant. A representative pool is therefore essential to the whole evaluation process.

This paper proposes a method to form the assessment set with the support of machine learning algorithms. Based on relevance judgments of relatively shallow pools, a ranking algorithm will attempt to give priority for relevant documents so that the assessment set can be fixed at a feasible size without skewing the system evaluation result. The judgment process is indeed kept as much subjective-free as possible: the first relevance feeback step is designed appropriately so that the assessor cannot give any bias towards any particular rank or any system, the learning process is completely transparent to the assessors and parameters of the ranking function are collection-tailored rather than exported from previous collections.

The method will then be evaluated on TREC ad-hoc collections. Results from our comprehensive experiment confirm that the qrels generated by our method are much more representative than those of the same size by the TREC method. The outcome qrels is substantially smaller, so much cheaper to produce than the official TREC qrels, yet their conclusions about system effectiveness are quite compatible.

The remaining of this paper is organized as follows. We review related work in Section 2. Section 3 presents the general framework of applying machine learning techniques to forming test collections. We also give a brief introduction

[1] http://trec.nist.gov

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 399–406,
New York, June 2006.

about RankBoost (Freund et al., 2003) and Ranking SVM (Joachims, 2002b), the two learning algorithms used in our experiment. Section 4 introduces data sets and experimental setup. Section 5 is dedicated to present experimental results according to different evaluation criteria. Precisely, Section 5.1 shows the capacity of small pools on identifying relevant documents and Section 5.2 illustrates their impact on system comparison; Section 5.3 presents statistical validation tests. We conclude and discuss perspectives in Section 6.

2 Related work

2.1 TREC methodology

Since the seminal work of test collection forming in 1975 (Sparck Jones and Van Rijsbergen, 1975), pooling has been outlined as the main approach to form the assessment set. The simple solution of round robin pooling from different systems proposed in that report has been adopted in most existing IR evaluation forums such as TREC, CLEF[2] or NTCIR[3]. For convenience, we will denote that strategy as TREC-style pooling. To have the assessment set, from submissions (restricted length $L = 1000$ for most TREC tracks), only n top documents per submission are pooled. Despite different technical tricks to control the final pool size such as gathering only principal runs or reducing the value of n, the assessment procedure is still quite time-consuming. In TREC 8 ad-hoc track, for example, despite limiting the pool depth n at 100 and gathering only 71 of 129 submissions, each assessor has to work with approximately 1737 documents per topic (precisely, between 1046 and 2992 documents). Assuming that it takes on average 30 seconds to read and judge a document, the whole judgment procedure for this topic set can therefore only terminate after a round-the-clock month. Meanwhile, a simple analysis on the ad-hoc collections from TREC-3 to TREC-8 revealed that there are on average 94% documents judged as non relevant. Since most of existing effectiveness measures do not take into account these non relevant documents, it would be bettter to not waste effort on judging non relevant documents provided that the quality of test collections is always conserved. Several advanced pool sampling methods have been proposed but due to some common drawbacks, none of them has been used in practice.

[2]http://www.clef-campaign.org/

[3]http://research.nii.ac.jp/ntcir/

2.2 Topic adaptive pooling

Zobel (Zobel, 1998) forms the shallow pools according to the TREC methodology. When there are enough judged documents (up to the set of 30 top documents per run in his experiment), an extrapolation function will then be estimated to predict the number of unpooled relevant documents. The idea is to judge more documents for topics that have high potential to have relevant documents else. Carterette and Allan (Carterette and Allan, 2005) have recently replaced that extrapolation function by statistical tests to distinguish runs. This method produced interesting empirical outcomes on TREC ad-hoc collections, lack however a sound theoretical basis and is clearly of very high complexity due to iterative statistical tests of every run pairs. Furthermore, this incremental/online pooling approach raises a major concern about the unbiasness requirement from the human judgment as the assessors know well that documents come later are of lower ranks, thus of lower relevance possibility.

2.3 System adaptive pooling

Cormack et al. (Cormack et al., 1998) propose the so-called Move-To-Front (MTF) heuristic to give priority for documents based on the corresponding system performance. In their experiment, the latter factor has been simply the number of non relevant documents this system has introduced to the pool since the last relevant document. Aslam et al. (Aslam et al., 2003) formulate this priority rule by adopting an online learning algorithm called Hedged (Freund and Schapire, 1997).

Our method relies on this idea of pushing ahead relevant documents by weighting retrieval systems. There are however two major differences. Whilst all aforementioned proposals favor *online* paradigm with a series of human interaction rounds, our method works in batch mode. We believe that the latter is more suitable for this task since it eliminates as much as possible the bias introduced by human assessor towards any document. Moreover, the batch mode enables us to exploit intuitively the inter-topic relationship what is not the case of on-

line paradigm. The second difference lies in the way of estimating the ranking function. It is widely accepted that machine learning techniques can deliver more reliable model on previously unseen data given much less training instances than any classical statistical techniques or expert rules can.

2.4 Generate *pseudo* assessment set

Several evaluation methodologies, especially for web search engines, have been proposed to evaluate systems *without* relevance judgment. These proposals can be grouped into two main categories. The first (Soboroff et al., 2001; Wu and Crestani, 2003; Nuray and Can, 2006) exploits *internal* information of submissions. The second (Can et al., 2004; Joachims, 2002a; Beitzel et al., 2003) benefits *external* resources such as document and query content, or those of web environment. We skip the second category since these resources are not available in generic situations.

Soboroff et al. (Soboroff et al., 2001) sample documents of a shallow pool (top ten documents returned by retrieval systems) based on statistics from past qrels. Wu and Crestani (Wu and Crestani, 2003), Nuray and Can (Nuray and Can, 2006) adopt metasearch strategies on document position. A certain number of top outcome documents will then be considered as relevant without any human verification. Different voting schemes have been tried in the two aformentioned papers. Their empirical experiment illustrated how the quality of these pseudo-qrels is sensible to the chosen voting scheme and to other parameters such as the pool depth or the diversity of systems used for fusion. They also confirm that *pseudo*-qrels are often unable to identify best systems.

In sum, the thorough literature review confirmed the importance of relevance assessment sets in IR evaluation yet the lack of an appropriate solution to have a reliable set given a moderate amount of judgment resource.

3 Machine learning based Pooling

3.1 General framework

Let M denote the topic set size available for the training purpose, N the number of participating systems, k_1 the pool depth to get the training data from any participating system and K the final pool size.

The training process consists of two main steps. Firstly, for each training topic, k_1 first documents of all N systems are gathered and the assessors are asked to assess all of these documents. Let $\mathcal{T}$ denote the outcome of this assessing step on all M topics. From the information of $\mathcal{T}$, a function f will then be learned which assigns to each document a value corresponding to its relevance degree for a given query.

At the usage time, for each given topic, the whole retrieved list of N systems will be fused. These documents will then be sorted in the decreasing order of their values according to f and the K top documents will be sent to the assessor for judgment. This last set of judgements will be the qrels used for the system evaluation.

In the training framework, it is clear that the second step plays the major role. An effective scoring function can substantially save the workload at the last assessment step. We will now focus on methods for estimating such scoring function.

3.2 Document ranking principle

The scoring function f can be estimated in different ways as seen in the last section. In this study, we adopt the learning-to-rank paradigm for estimating this scoring function. The principle of document ranking will be sketched in this section. The next sub-section will introduce the two specific ranking algorithms used in our experiment.

A ranking algorithm aims at estimating a function which describes correctly all partial orders inside a set of elements. An ideal ranking in information retrieval must be able to place all relevant documents above non relevant ones for a given topic. The problem can be described as follows. For each topic, the document collection is decomposed into two disjoint sets $\mathcal{S}_+$ and $\mathcal{S}_-$ for relevant (non relevant respectively) documents, R and NR are their cardinality. A ranking function $H(d)$ assigns to each document d of the document collection a score value. We seek for a function $H(d)$ so that the document ranking generated from the scores respect the relevance relationship, that is any relevant document has a higher score than any non relevant one. Let "$d \triangleright d'$" signify that d is ranked higher than d'. The learning

objective can therefore be stated as follows.

$$d_+ \rhd d_- \Leftrightarrow H(d_+) > H(d_-), \forall (d_+, d_-) \in \mathcal{S}_+ \times \mathcal{S}_-$$

There are different ways to measure the ranking error of a scoring function h. The natural criterion might be the proportion of misordered pairs (a relevant document is below a non relevant one) over the total pair number $R.NR$. This criterion is an estimate of the probability of misordering a pair $P(d_- \rhd d_+)$.

$$\mathbf{RLoss}(H) = \sum_{\substack{d_+ \in \mathcal{S}_+ \\ d_- \in \mathcal{S}_-}} D(d_+, d_-) [\![d_- \rhd d_+]\!] \quad (1)$$

$$= \sum_{\forall (d_+, d_-)} D(d_+, d_-) [\![H(d_-) > H(d_+)]\!] \quad (2)$$

where $[\![\phi]\!]$ is 1 if ϕ holds, 0 otherwise; $D(d_+, d_-)$ describes the importance of the pair in consideration, it will be uniform $\left(\frac{1}{R.NR}\right)$ if the information is unknown.

In practice, we have to average RLoss over the training topic set. This can be done by either *macro*-averaging at topic level or *micro*-averaging at document pair level. For presentation simplification, this operation has been implicit.

3.3 Discriminative ranking algorithms

Since RLoss is neither continuous nor differentiable, its direct use as a training criterion raises practical difficulties. Also, in order to provide reliable predictions on previously unseen data, the prediction error of the learning function has to be bounded with a significant confidence. For both practical and theoretical reasons, RLoss is then often approximated by a smooth error function.

In this study, we will explore the performance of two ranking algorithms, they are RankBoost (Freund et al., 2003) and Ranking SVM (Joachims, 2002b). As far as we know, these algorithms are actually among a few state-of-the-art ranking learning algorithms whose convergence and generalization properties have been theoretically proved (Freund et al., 2003; Joachims, 2002b; Clémençon et al., 2005).

3.3.1 RankBoost

RankBoost (aka RBoost) (Freund et al., 2003) returns a scoring function for each document d by minimizing the following *exponential* upper bound of the ranking error RLoss (Eq. (2)):

$$\mathbf{ELoss}(H) = \sum_{(d_+, d_-)} D(d_+, d_-) e^{H(d_-) - H(d_+)} \quad (3)$$

This is an iterative algorithm like all other boosting methods (Freund and Schapire, 1997). The global ranking function of a document d is a linear combination of all base functions $H(d) = \sum_{t=1}^{T} \alpha_t h_t(d)$. At each iteration t, a new training data sample is generated by putting more weight $D(.,.)$ on difficult pairs (d_+, d_-). A scoring function h_t is proposed (it can even be chosen among the features used to describe documents) and the weight α_t is estimated in order to minimize the ELoss at that iteration.

RBoost has virtues particularly fitting the pooling task. First, it can operate on relative values. Second, it does not impose any independence assumption between combined systems. Finally, in the case of binary relevance judgment which usually occurs in IR, there is an efficient implementation of RBoost whose complexity is linear in terms of the training instance number (cf: the original text (Freund et al., 2003)).

3.3.2 Ranking SVM

Ranking SVM (Joachims, 2002b), rSVM for short, is a straightforward adaptation of the max-margin principle (Vapnik, 2000) to pairwise object ranking. The score function is often assumed to be linear in some feature space, that is $H(d) = \mathbf{w}^T \mathbf{\Psi}(d)$ where $\mathbf{w}$ is the vector of weights to be estimated and $\mathbf{\Psi}$ is a feature mapping. The max-margin approach minimizes the following approximation of RLoss:

$$\mathbf{rSVMLoss}(H) = \max \left\{ 1 + \left(H(d_-) - H(d_+) \right), 0 \right\} \quad (4)$$

for all pairs (d_+, d_-) while at the same time controlling the complexity of function space described via the norm of vector $\mathbf{w}$ for generalization objective.

Notice that rSVM does not explicitly support rank values as does RBoost. Nevertheless, we will see later that the discriminative nature allows rSVM to work quite well on features merely deduced from rank values. Its behavior difference is in fact ignorable in comparison with RBoost.

4 Experimental setup

Our method is general enough to be applicable to any ad-hoc retrieval information task where pooling

could be useful. In this paper, we will however focus on TREC traditional ad-hoc retrieval collections. Experiments have been performed on the three corpora TREC-6, TREC-7 and TREC-8. Statistics about the number of runs, of judgments, of relevant documents are shown in Tab. 1. Due to limit of space, we will detail results on the TREC-8 case and only mention the results on the two others.

	#runs	#judgments	#rel. docs	Depth-5
TREC 6	79	1445.4	92.2	144.3
TREC 7	103	1606.9	93.5	114.6
TREC 8	129	1736.6	94.6	143.4

Table 1: Information about three TREC ad-hoc collections. The three last columns are averaged over the topic set size (50 topics/collection).

Training data is gathered from the top five answers of each run. The pool depth of five has been arbitrarily chosen to have both sufficient training data and to eliminate potential bias from assessors towards a particular system or towards early identified answers while judging a shallow pool. Furthermore, this training data set is large enough for testing the ranking algorithm efficiency.

Each document is described by an N-dimensional feature vector where N is the number of participating systems. The j^{th} feature value for a document is a function of its position in the retrieved list, ties are arbitrary broken. A document at rank i is assigned a feature value of $(L + 1 - i)$ where L is the TREC limit of submission run (L is usually set up at 1000). Documents outside submission runs receive the zero feature value (i.e. it is assumed to be at rank $(L + 1)$). For implementation speed, the input for rSVM is further scaled down to the interval $[0, 1]$.

Due to the small topic set size, we use a *leave-one-out* training strategy: a model will be trained for each topic by using judgments of all other topics. The training data set size is presented in the last column of Tab. 1. The workload for training dataset does not exceed the effort for assessing 5 topics in the full pool of TREC.

We employ SVM^{light} package[4] for rSVM. We adopt the efficient RBoost version for binary feedback and binary base functions h_t (cf. (Freund et al., 2003)), boosting is iterated 100 times and we impose positive weighting for all coefficients α_t.

The non-interpolated average precision (MAP) has been chosen to measure system performance[5]. This metric has been shown to be highly stable and reliable with both small topic set size (Buckley and Voorhees, 2000) and very large document collections (Hawking and Robertson, 2003).

RBoost and rSVM pools will be compared to the TREC-style pools of the same size. We also include "local MTF" (Cormack et al., 1998) in the experiment. The "global MTF" has been shown to slightly outperform the local version in the aforementioned paper. However, we believe that the global mode is merely for demonstration but unlikely practical of online judgment since it insists that all queries are judged simultaneously with a strict synchronisation among all assessors. Hereafter, for simplicity, the TREC-style pool of the first n documents retrieved by each submission will be denoted by Depth-n, the equivalent pool (with the same average final pool size m over the topic set) produced by RBoost, rSVM or MTF will be RBoost-m, rSVM-m or MTF-m respectively. In all figures in the next section, the abscissa denotes the pool size m and values of n will be present along the Depth-n curve.

5 Experimental results

This section will examine small pools produced either by the TREC method or by RBoost/rSVM/MTF from two angles: their pooling performance and their influence on system comparison result.

5.1 Identify *relevant* documents

Fig. 1 shows the ratio of relevant documents retrieved by different pooling methods (i.e. the recall). The curves obtained by RBoost and rSVM are quite similar and much higher than that by TREC methodology. The curve of MTF is in the middle of RBoost/rSVM and Depth-n at the beginning and then catches that of RBoost at the pools of about 600 documents.

[4] http://svmlight.joachims.org

[5] http://trec.nist.gov/trec_eval/

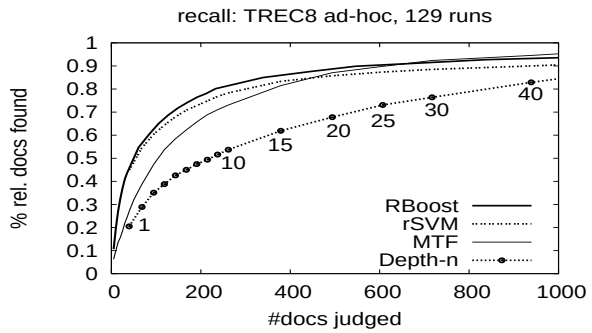


Figure 1: Along the incrementally enlarged pools: *relevant* documents identified in comparison with the full assessment set.

5.2 Correlation of system rankings

Once the pool is obtained by a given method, the assessor will give relevance judgment for all documents of that pool, called qrels for the outcome. This qrels will be used as the ground truth to measure effectiveness of a retrieval system.

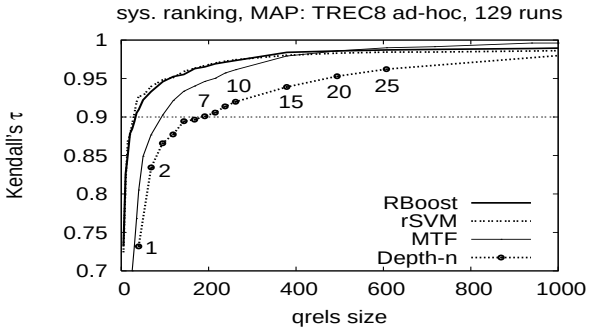


Figure 2: Kendall's τ correlation of system ranking according to different qrels methods in comparison with that produced by the full assessment set.

The simplest way to compare different systems is to sort them by the decreasing effectiveness values. The correlation of each two system rankings will then be quantified through a correlation statistic. In this study, we follow TREC convention (Buckley and Voorhees, 2004; Carterette and Allan, 2005), that is taking the 0.9 value of Kendall's τ as the sufficient threshold to conclude that the difference of two system rankings is ignorable. We compare here the system ranking obtained by the official TREC qrels with those by Depth-*n* where *n* varies from 1 to 100. We then replace Depth-*n* by RBoost-*m*, rSVM-*m* and MTF-*m*. The results are shown in Fig. 2 for TREC-8 and in Tab. 2 for the 7 first pool depths. We observe from the figure the similar order of pooling methods as seen in the previous section. The MTF curve meets those of RBoost and rSVM from qrels of more than 400 documents. The results obtained on the two collections of TREC-6 and TREC-7 are in line with those observed on TREC-8 (Tab. 2).

It is clear that system ranking correlation quantified by any rank correlation statistics provides necessary but not sufficient information about system comparison. Ranking systems by their sample means is indeed the simplest way with at least two implicit assumptions. First, runs have *similar variances*, this usually does not hold in practice even after discarding poorest runs. Second, all run swaps have the same importance without taking into account their *statistically significant difference* and their positions in the system ranking. In practice, swap of adjacent systems does not make much sense if they are not significantly different to each other according to statstical tests. The next section will be devoted to further statistical validations.

5.3 Statistical Validations

5.3.1 Significant difference detection

We register for a given qrels all system pairs which are significantly different on the topic set. The quality of a qrels can be measured by the similarity of this significant difference detection in comparison with that obtained by the official TREC qrels. We carry out the paired t-test for each pair of runs with 95% significance level. The recall and the false alarm rate of these detections are shown in Fig. 3. In terms of recall, RBoost and rSVM qrels are much more better than its TREC-style counterparts and MTF is in the middle. In terms of false alarm rate, there are some changes concerning rSVM and MTF. Precisely, rSVM at small qrels of less than 100 documents is the best whilst that is MTF qrels of more than 150 documents.

5.3.2 Tukey grouping

This multicomparison test[6] aims to group runs based on their statistical difference. We concentrate

[6]IR-STAT-PAK (Tague-Sutcliffe and Blustein, 1995)

	TREC-6 (79 sys.)					TREC-7 (103 sys.)					TREC-8 (129 sys.)				
n	m	D-*n*	MTF	SVM	RBst	m	D-*n*	MTF	SVM	RBst	m	D-*n*	MTF	SVM	RBst
1	37	.835	.843	.888	**.914**	32	.788	.809	.888	.891	40	.733	.805	**.927**	**.909**
2	66	.875	.899	**.925**	.934	56	.831	.890	**.920**	**.922**	68	.829	.877	.939	.933
3	93	.892	**.925**	.939	.956	76	.851	**.918**	.931	.935	95	.864	**.903**	.948	.946
4	118	**.903**	.940	.949	.967	95	.876	.926	.935	.947	119	.877	.921	.951	.953
5	144	.907	.949	.958	.972	115	.884	.936	.942	.954	143	.896	.933	.959	.955
6	170	.915	.953	.961	.974	133	.894	.942	.951	.957	168	.898	.940	.963	.963
7	195	.925	.959	.967	.977	152	**.903**	.950	.956	.962	191	**.901**	.946	.967	.966

Table 2: Kendall's τ obtained on small qrels. D-n: TREC-style Depth-n qrels, SVM: rSVM-m; RBst: RBoost-m.

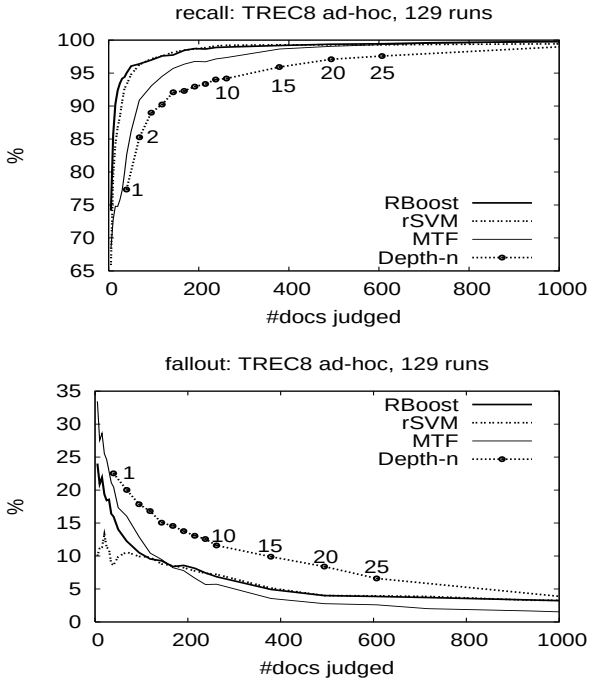


Figure 3: Comparing qrels of RBoost-*m*, rSVM-*m*, MTF-*m* and Depth-*n* in terms of pairs of significantly different systems: recall (top) and false alarm rate (bottom)

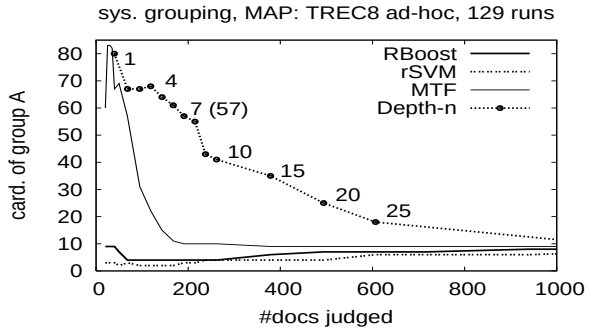


Figure 4: Cardinality of group A (95% confidence level) after the arcsine-root data transformation.

particularly on *the top group*, called group A which consists of runs on which there is *not enough evidence* to conclude that they are statistically significantly worse than *the top run*. In practice, this figure will be meaningful if it is around 10 (one often says about the top 10 runs). It will however become meaningless if the group A is too large, for example contains more than half of systems in consideration. Note that Tukey test relies on the assumption of Equality of Variances. This requirement can not be completely satisfied in practice, even after some data transformation such as arcsine-root or using rank values.

The size of group A on TREC-8 collection is shown in Fig. 4. We observe from that figure the stability of the two curves of RBoost and rSVM, this implies that the two qrels RBoost-35 and rSVM-35 which have both satisfied the 0.9 requirement of Kendall's τ can replace the official TREC qrels. The effort saving is therefore a factor of 50 (if ignoring the cost of training data set preparation) and of 10.5 otherwise. MTF needs qrels of at least 168 documents to produce comparable group A's with that of the official TREC qrels. The Depth-*n* pools however should not be recommended with less than 1000 documents in total (i.e. pooling more than 40 top documents per run).

6 Conclusions and Discussion

This study has well illustrated that two algorithms of RBoost and rSVM are quite suitable for qrels construction task. The final qrels are not only small enough to ask for human judgment but also result in reliable conclusion about system effectiveness in

comparison with the counterpart of TREC methodology and that of MTF.

It is necessary to include other metasearch methods for further study. This will allow us to validate not only the impact of the metasearch training principle based on pairwise ranking error RLoss but also the capacity of automatic feature selection of the two ranking algorithms used in this paper.

This method needs to be further verified on challenging ad-hoc retrieval scenarios such as Terabyte, Web Topic Distillation or Robust Tracks in TREC context. The hardness of these scenarios involves two main issues. First, the number of document judged relevant varies largely across the whole topic set. Second, some topics might even have no relevant document in shallow pools. These matter any statistical inference on shallow pools.

Acknowledgement The authors thank M.-R. Amini, B. Piwowarski, J. Zobel and the anonymous reviewers for their thorough comments. We acknowledge NIST to make accessible the TREC submissions. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. The publication only reflects the authors' views.

References

[Aslam et al.2003] J.A. Aslam, V. Pavlu, and R. Savell. 2003. A unified model for metasearch, pooling, and system evaluation. In *Proc. CIKM'03*.

[Beitzel et al.2003] S. M. Beitzel, E. C. Jensen, A. Chowdhury, and D. Grossman. 2003. Using titles and category names from editor-driven taxonomies for automatic evaluation. In *Proc. CIKM'03*.

[Buckley and Voorhees2000] C. Buckley and E.M. Voorhees. 2000. Evaluating evaluation measure stability. In *Proc. SIGIR'00*.

[Buckley and Voorhees2004] C. Buckley and E.M. Voorhees. 2004. Retrieval evaluation with incomplete information. In *Proc. SIGIR'04*.

[Can et al.2004] F. Can, R. Nuray, and A. B. Sevdik. 2004. Automatic performance evaluation of web search engines. *Info. Process. Management*, 40(3):495–514, May.

[Carterette and Allan2005] B. Carterette and J. Allan. 2005. Incremental Test Collections. In *CIKM'05*.

[Clémençon et al.2005] S. Clémençon, G. Lugosi, and N. Vayatis. 2005. Ranking and scoring using empirical risk minimization. In *Proc. COLT'05*.

[Cormack et al.1998] G.V. Cormack, Christopher R. Palmer, and C.L.A. Clarke. 1998. Efficient construction of large test collections. In *Proc. SIGIR'98*.

[Freund and Schapire1997] Y. Freund and R.E. Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Compt. Sys. Sci.*, 55(1):119–139, August.

[Freund et al.2003] Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer. 2003. An efficient boosting algorithm for combining preferences. *J. Mach. Learning Res.*, 4:933–969, November.

[Hawking and Robertson2003] D. Hawking and S. Robertson. 2003. On collection size and retrieval effectiveness. *Information Retrieval*, 6(1):99–105.

[Joachims2002a] Th. Joachims. 2002a. Evaluating retrieval performance using clickthrough data. In *Proc. SIGIR wshop on Math./Formal Methods in IR*.

[Joachims2002b] Th. Joachims. 2002b. Optimizing search engines using clickthrough data. In *KDD'02*.

[Nuray and Can2006] R. Nuray and F. Can. 2006. Automatic ranking of information retrieval systems using data fusion. *Info. Process. Management*, 42(3):595–614, May.

[Soboroff et al.2001] I. Soboroff, Ch. Nicholas, and P. Cahan. 2001. Ranking Retrieval Systems without Relevance Judgments. In *Proc. SIGIR'01*.

[Sparck Jones and Van Rijsbergen1975] K. Sparck Jones and C. J. Van Rijsbergen. 1975. Report on the need for and provision of an ideal information retrieval test collection. Technical Report 5266, Computer Lab., Univ. Cambridge.

[Tague-Sutcliffe and Blustein1995] J. Tague-Sutcliffe and J. Blustein. 1995. A statistical analysis of the TREC-3 data. In *Proc. TREC-3*.

[Vapnik2000] N. V. Vapnik. 2000. *The Nature of Statistical Learning Theory*. Springer-Verlag.

[Voorhees and Harman1999] E.M. Voorhees and D. Harman. 1999. Overview of the Eighth Text REtrieval Conference (TREC-8). In *Proc. TREC 8*.

[Wu and Crestani2003] Sh. Wu and F. Crestani. 2003. Methods for Ranking Information Retrieval Systems Without Relevance Judgements. In *SAC'03*.

[Zobel1998] J. Zobel. 1998. How reliable are the results of large-scale information retrieval experiments? In *Proc. SIGIR'98*.

Language Model Information Retrieval with Document Expansion

Tao Tao, Xuanhui Wang, Qiaozhu Mei, ChengXiang Zhai
Department of Computer Science
University of Illinois at Urbana_Champaign

Abstract

Language model information retrieval depends on accurate estimation of document models. In this paper, we propose a document expansion technique to deal with the problem of insufficient sampling of documents. We construct a probabilistic neighborhood for each document, and expand the document with its neighborhood information. The expanded document provides a more accurate estimation of the document model, thus improves retrieval accuracy. Moreover, since document expansion and pseudo feedback exploit different corpus structures, they can be combined to further improve performance. The experiment results on several different data sets demonstrate the effectiveness of the proposed document expansion method.

1 Introduction

Information retrieval with statistical language models (Lafferty and Zhai, 2003) has recently attracted much more attention because of its solid theoretical background as well as its good empirical performance. In this approach, queries and documents are assumed to be sampled from hidden generative models, and the similarity between a document and a query is then calculated through the similarity between their underlying models.

Clearly, good retrieval performance relies on the accurate estimation of the query and document models. Indeed, smoothing of document models has been proved to be very critical (Chen and Goodman, 1998; Kneser and Ney, 1995; Zhai and Lafferty, 2001b). The need for smoothing originated from the zero count problem: when a term does not occur in a document, the maximum likelihood estimator would give it a zero probability. This is unreasonable because the zero count is often due to insufficient sampling, and a larger sample of the data would likely contain the term. Smoothing is proposed to address the problem.

While most smoothing methods utilize the global collection information with a simple interpolation (Ponte and Croft, 1998; Miller et al., 1999; Hiemstra and Kraaij, 1998; Zhai and Lafferty, 2001b), several recent studies (Liu and Croft, 2004; Kurland and Lee, 2004) have shown that local corpus structures can be exploited to improve retrieval performance. In this paper, we further study the use of local corpus structures for document model estimation and propose to use document expansion to better exploit local corpus structures for estimating document language models.

According to statistical principles, the accuracy of a statistical estimator is largely determined by the sampling size of the observed data; a small data set generally would result in large variances, thus can not be trusted completely. Unfortunately, in retrieval, we often have to estimate a model based on a single document. Since a document is a small sample, our estimate is unlikely to be very accurate.

A natural improvement is to enlarge the data sample, ideally in a document-specific way. Ideally, the enlarged data sample should come from the same original generative model. In reality, however, since

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 407–414,
New York, June 2006.

the underlying model is unknown to us, we would not really be able to obtain such extra data. The essence of this paper is to use *document expansion* to obtain high quality extra data to enlarge the sample of a document so as to improve the accuracy of the estimated document language model. Document expansion was previously explored in (Singhal and Pereira, 1999) in the context of the vector space retrieval model, mainly involving selecting more terms from similar documents. Our work differs from this previous work in that we study document expansion in the language modeling framework and implement the idea quite differently.

Our main idea is to augment a document probabilistically with potentially all other documents in the collection that are similar to the document. The probability associated with each neighbor document reflects how likely the neighbor document is from the underlying distribution of the original document, thus we have a "probabilistic neighborhood", which can serve as "extra data" for the document for estimating the underlying language model. From the viewpoint of smoothing, our method extends the existing work on using clusters for smoothing (Liu and Croft, 2004) to allow each document to have its own cluster for smoothing.

We evaluated our method using six representative retrieval test sets. The experiment results show that document expansion smoothing consistently outperforms the baseline smoothing methods in all the data sets. It also outperforms a state-of-the-art clustering smoothing method. Analysis shows that the improvement tends to be more significant for short documents, indicating that the improvement indeed comes from the improved estimation of the document language model, since a short document presumably would benefit more from the neighborhood smoothing. Moreover, since document expansion and pseudo feedback exploit different corpus structures, they can be combined to further improve performance. As document expansion can be done in the indexing stage, it is scalable to large collections.

2 Document Expansion Retrieval Model

2.1 The KL-divergence retrieval model

We first briefly review the KL-divergence retrieval model, on which we will develop the document expansion technique. The KL-divergence model is a representative state-of-the-art language modeling approach for retrieval. It covers the basic language modeling approach (i.e., the query likelihood method) as a special case and can support feedback more naturally.

In this approach, a query and a document are assumed to be generated from a unigram query language model Θ_Q and a unigram document language model Θ_D, respectively. Given a query and a document, we would first compute an estimate of the corresponding query model ($\hat{\Theta}_Q$) and document model ($\hat{\Theta}_D$), and then score the document w.r.t. the query based on the KL-divergence of the two models (Lafferty and Zhai, 2001):

$$D(\hat{\Theta}_Q \,||\, \hat{\Theta}_d) = \sum_{w \in V} p(w|\hat{\Theta}_Q) \times \log \frac{p(w|\hat{\Theta}_Q)}{p(w|\hat{\Theta}_d)}.$$

where V is the set of all the words in our vocabulary. The documents can then be ranked according to the ascending order of the KL-divergence values.

Clearly, the two fundamental problems in such a model are to estimate the query model and the document model, and the accuracy of our estimation of these models would affect the retrieval performance significantly. The estimation of the query model can often be improved by exploiting the local corpus structure in a way similar to pseudo-relevance feedback (Lafferty and Zhai, 2001; Lavrenko and Croft, 2001; Zhai and Lafferty, 2001a). The estimation of the document model is most often done through smoothing with the global collection language model (Zhai and Lafferty, 2001b), though recently there has been some work on using clusters for smoothing (Liu and Croft, 2004). Our work is mainly to extend the previous work on document smoothing and improve the accuracy of estimation by better exploiting the local corpus structure. We now discuss all these in detail.

2.2 Smoothing of document models

Given a document d, the simplest way to estimate the document language model is to treat the document as a sample from the underlying multinomial word distribution and use the maximum likelihood estimator: $P(w|\hat{\Theta}_d) = \frac{c(w,d)}{|d|}$, where $c(w,d)$ is the count of word w in document d, and $|d|$ is the

length of d. However, as discussed in virtually all the existing work on using language models for retrieval, such an estimate is problematic and inaccurate; indeed, it would assign zero probability to any word not present in document d, causing problems in scoring a document with query likelihood or KL-divergence (Zhai and Lafferty, 2001b). Intuitively, such an estimate is inaccurate because the document is a small sample.

To solve this problem, many different smoothing techniques have been proposed and studied, usually involving some kind of interpolation of the maximum likelihood estimate and a global collection language model (Hiemstra and Kraaij, 1998; Miller et al., 1999; Zhai and Lafferty, 2001b). For example, Jelinek-Mercer(JM) and Dirichlet are two commonly used smoothing methods (Zhai and Lafferty, 2001b). JM smoothing uses a fixed parameter λ to control the interpolation:

$$P(w|\hat{\Theta}_d) = \lambda \frac{c(w,d)}{|d|} + (1-\lambda)P(w|\Theta_C),$$

while the Dirichlet smoothing uses a document-dependent coefficient (parameterized with μ) to control the interpolation:

$$P(w|\hat{\Theta}_d) = \frac{c(w,d) + \mu P(w|\Theta_C)}{|d| + \mu}.$$

Here $P(w|\Theta_C)$ is the probability of word w given by the collection language model Θ_C, which is usually estimated using the whole collection of documents C, e.g., $P(w|\Theta_C) = \frac{\sum_{d \in C} c(d,w)}{\sum_{d \in C} |d|}$.

2.3 Cluster-based document model (CBDM)

Recently, the cluster structure of the corpus has been exploited to improve language models for retrieval (Kurland and Lee, 2004; Liu and Croft, 2004). In particular, the cluster-based language model proposed in (Liu and Croft, 2004) uses clustering information to further smooth a document model. It divides all documents into K different clusters ($K = 1000$ in their experiments). Both cluster information and collection information are used to improve the estimate of the document model:

$$\begin{aligned} P(w|\hat{\Theta}_d) &= \lambda \frac{c(w,d)}{|d|} + (1-\lambda) \\ &\quad \times [\beta P(w|\Theta_{L_d}) + (1-\beta)P(w|\Theta_C)], \end{aligned}$$

where Θ_{L_d} stands for document d's cluster model and λ and β are smoothing parameters. In this clustering-based smoothing method, we first smooth a cluster model with the collection model using Dirichlet smoothing, and then use smoothed cluster model as a new reference model to further smooth the document model using JM smoothing; empirical results show that the added cluster information indeed enhances retrieval performance (Liu and Croft, 2004).

2.4 Document expansion

From the viewpoint of data augmentation, the clustering-based language model can be regarded as "expanding" a document with more data from the cluster that contains the document. This is intuitively better than simply expanding every document with the same collection language model as in the case of JM or Dirichlet smoothing. Looking at it from this perspective, we see that, as the "extra data" for smoothing a document model, the cluster containing the document is often not optimal. Indeed, the purpose of clustering is to group similar documents together, hence a cluster model represents well the overall property of *all* the documents in the cluster. However, such an average model is often not accurate for smoothing each individual document. We illustrate this problem in Figure 1(a), where we show two documents d and a in cluster D. Clearly the generative model of cluster D is more suitable for smoothing document a than document d. In general, the cluster model is more suitable for smoothing documents close to the centroid, such as a, but is inaccurate for smoothing a document at the boundary, such as d.

To achieve optimal smoothing, each document should ideally have its own cluster centered on the document, as shown in Figure 1(b). This is precisely what we propose – expanding each document with a probabilistic neighborhood around the document and estimate the document model based on such a virtual, expanded document. We can then apply any simple interpolation-based method (e.g., JM or Dirichlet) to such a "virtual document" and treat the word counts given by this "virtual document" as if they were the original word counts.

The use of neighborhood information is worth more discussion. First of all, neighborhood is not a

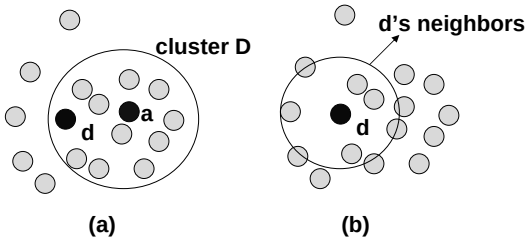


Figure 1: Clusters, neighborhood, and document expansion

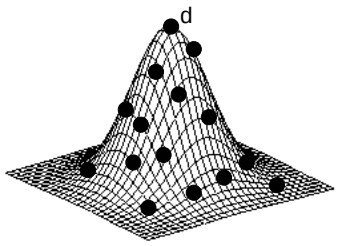


Figure 2: Normal distribution of confidence values.

clearly defined concept. In the narrow sense, only a few documents close to the original one should be included in the neighborhood, while in the wide sense, the whole collection can be potentially included. It is thus a challenge to define the neighborhood concept reasonably. Secondly, the assumption that neighbor documents are sampled from the same generative model as the original document is not completely valid. We probably do not want to trust them so much as the original one. We solve these two problems by associating a confidence value with every document in the collection, which reflects our belief that the document is sampled from the same underlying model as the original document. When a document is close to the original one, we have high confidence, but when it is farther apart, our confidence would fade away. In this way, we construct a probabilistic neighborhood which can potentially include all the documents with different confidence values. We call a language model based on such a neighborhood *document expansion language model* (DELM).

Technically, we are looking for a new *enlarged* document d' for each document d in a text collection, such that the new document d' can be used to estimate the hidden generative model of d more accurately. Since a good d' should presumably be based on both the original document d and its neighborhood $N(d)$, we define a function ϕ:

$$d' = \phi(d, N(d)). \quad (1)$$

The precise definition of the neighborhood concept $N(d)$ relies on the distance or similarity between each pair of documents. Here, we simply choose the commonly used cosine similarity, though other choices may also be possible. Given any two document models X and Y, the cosine similarity is defined as:

$$sim(X, Y) = \frac{\sum_i x_i \times y_i}{\sqrt{\sum_i (x_i)^2 \times \sum_i (y_i)^2}}.$$

To model the uncertainty of neighborhood, we assign a confidence value $\gamma_d(b)$ to every document b in the collection to indicate how strongly we believe b is sampled from d's hidden model. In general, $\gamma_d(b)$ can be set based on the similarity of b and d – the more similar b and d are, the larger $\gamma_d(b)$ would be. With these confidence values, we construct a probabilistic neighborhood with every document in it, each with a different weight. The whole problem is thus reduced to how to define $\gamma_d(b)$ exactly.

Intuitively, an exponential decay curve can help regularize the influence from remote documents. We therefore want $\gamma_d(b)$ to satisfy a normal distribution centered around d. Figure 2 illustrates the shape of this distribution. The black dots are neighborhood documents centered around d. Their probability values are determined by their distances to the center. We fortunately observe that the cosine similarities, which we use to decide the neighborhood, are roughly of this decay shape. We thus use them directly without further transformation because that would introduce unnecessary parameters. We set $\gamma_d(b)$ by normalizing the cosine similarity scores :

$$\gamma_d(b) = \frac{sim(d, b)}{\sum_{b' \in C - \{d\}} sim(d, b')}.$$

Function ϕ serves to balance the confidence between d and its neighborhood $N(d)$ in the model estimation step. Intuitively, a shorter document is less sufficient, hence needs more help from its neighborhood. Conversely, a longer one can rely more on itself. We use a parameter α to control this balance. Thus finally, we obtain a pseudo document d' with

the following pseudo term count:

$$c(w,d') = \alpha c(w,d) + (1-\alpha) \times \sum_{b \in C-\{d\}} (\gamma_d(b) \times c(w,b)),$$

We hypothesize that, in general, Θ_d can be estimated more accurately from d' rather than d itself because d' contains more complete information about Θ_d. This hypothesis can be tested by by comparing the retrieval results of applying any smoothing method to d with those of applying the same method to d'. In our experiments, we will test this hypothesis with both JM smoothing and Dirichlet smoothing.

Note that the proposed document expansion technique is quite general. Indeed, since it transforms the original document to a potentially better "expanded document", it can presumably be used together with any retrieval method, including the vector space model. In this paper, we focus on evaluating this technique with the language modeling approach.

Because of the decay shape of the neighborhood and for the sake of efficiency, we do not have to actually use all documents in $C-\{d\}$. Instead, we can safely cut off the documents on the tail, and only use the top M closest neighbors for each document. We show in the experiment section that the performance is not sensitive to the choice of M when M is sufficiently large (for example 100). Also, since document expansion can be done completely offline, it can scale up to large collections.

3 Experiments

We evaluate the proposed method over six representative TREC data sets (Voorhees and Harman, 2001): AP (Associated Press news 1988-90), LA (LA Times), WSJ (Wall Street Journal 1987-92), SJMN (San Jose Mercury News 1991), DOE (Department of Energy), and TREC8 (the ad hoc data used in TREC8). Table 1 shows the statistics of these data.

We choose the first four TREC data sets for performance comparison with (Liu and Croft, 2004). To ensure that the comparison is meaningful, we use identical sources (after all preprocessing). In addition, we use the large data set TREC8 to show that our algorithm can scale up, and use DOE because its documents are usually short, and our previous experience shows that it is a relatively difficult data set.

	#document	queries	#total qrel
AP	242918	51-150	21819
LA	131896	301-400	2350
WSJ	173252	51-100 and 151-200	10141
SJMN	90257	51-150	4881
TREC8	528155	401-450	4728
DOE	226087	DOE queries	2047

Table 1: Experiment data sets

3.1 Neighborhood document expansion

Our model boils down to a standard query likelihood model when no neighborhood document is used. We therefore use two most commonly used smoothing methods, JM and Dirichlet , as our baselines. The results are shown in Table 2, where we report both the mean average precision (MAP) and precision at 10 documents. JM and Dirichlet indicate the standard language models with JM smoothing and Dirichlet smoothing respectively, and the other two are the ones combined with our document expansion. For both baselines, we tune the parameters (λ for JM, and μ for Dirichlet) to be optimal. We then use the same values of λ or μ without further tuning for the document expansion runs, which means that the parameters may not necessarily optimal for the document expansion runs. Despite this disadvantage, we see that the document expansion runs significantly outperform their corresponding baselines, with more than 15% relative improvement on AP. The parameters M and α were set to 100 and 0.5, respectively.

To understand the improvement in more detail, we show the precision values at different levels of recall for the AP data in Table 3. Here we see that our method significantly outperforms the baseline at every precision point.

In our model, we introduce two additional parameters: M and α. We first examine M here, and then study α in Section 3.3. Figure 3 shows the performance trend with respect to the values of M. The x-axis is the values of M, and the y-axis is the non-interpolated precision averaging over all 50 queries. We draw two conclusions from this plot: (1) Neighborhood information improves retrieval accuracy; adding more documents leads to better retrieval results. (2) The performance becomes insensitive to

Data		JM	DELM+JM (impr. %)		Dirichlet	DELM + Diri.(impr. %)	
AP	AvgPrec	0.2058	0.2405	(16.8%***)	0.2168	0.2505	(15.5%***)
	P@10	0.3990	0.4444	(11.4%***)	0.4323	0.4515	(4.4%**)
DOE	AvgPrec	0.1759	0.1904	(8.3%***)	0.1804	0.1898	(5.2%**)
	P@10	0.2629	0.2943	(11.9%*)	0.2600	0.2800	(7.7%*)
TREC8	AvgPrec	0.2392	0.2539	(6.01%**)	0.2567	0.2671	(4.05%*)
	P@10	0.4300	0.4460	(3.7%)	0.4500	0.4740	(5.3%*)

Table 2: Comparisons with baselines. *,**,*** indicate that we accept the improvement hypothesis by Wilcoxon test at significance level 0.1, 0.05, 0.01 respectively.

AP, TREC queries 51-150			
	Dirichlet	DELM+Diri	Improvement(%)
Rel.	21819	21819	
Rel.Retr.	10126	10917	7.81% ***
Prec.			
0.0	0.6404	0.6605	3.14% *
0.1	0.4333	0.4785	10.4% ***
0.2	0.3461	0.3983	15.1% ***
0.3	0.2960	0.3496	18.1% ***
0.4	0.2436	0.2962	21.6% ***
0.5	0.2060	0.2418	17.4% ***
0.6	0.1681	0.1975	17.5% ***
0.7	0.1290	0.1580	22.5% ***
0.8	0.0862	0.1095	27.0% **
0.9	0.0475	0.0695	46.3% **
1.0	0.0220	0.0257	16.8%
ave.	0.2168	0.2505	15.5% ***

Table 3: PR curve on AP data. *,**,*** indicate that we accept the improvement hypothesis by Wilcoxon test at significant level 0.1, 0.05, 0.01 respectively.

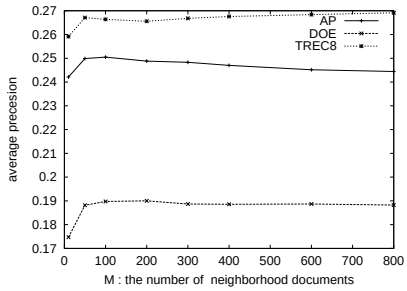


Figure 3: Performance change with respect to M

	CBDM	DELM+Diri.	improvement(%)
AP	0.2326	0.2505	7.7%
LA	0.2590	0.2655	2.5%
WSJ	0.3006	0.3113	3.6%
SJMN	0.2171	0.2266	4.3%

Table 4: Comparisons with CBDM.

M when M is sufficiently large, namely 100. The reason is twofold: First, since the neighborhood is centered around the original document, when M is large, the expansion may be evenly magnified on all term dimensions. Second, the exponentially decaying confidence values reduce the influence of remote documents.

3.2 Comparison with CBDM

In this section, we compare the CBDM method using the model performing the best in (Liu and Croft, 2004)[1]. Furthermore, we also set Dirichlet prior parameter $\mu = 1000$, as mentioned in (Liu and Croft, 2004), to rule out any potential influence of Dirichlet smoothing.

Table 4 shows that our model outperforms CBDM in MAP values on four data sets; the improvement presumably comes from a more principled way of exploiting corpus structures. Given that clustering can at least capture the local structure to some extent, it should not be very surprising that the improvement of document expansion over CBDM is much less than that over the baselines.

Note that we cannot fulfill Wilcoxon test because of the lack of the individual query results of CBDM.

3.3 Impact on short documents

Document expansion is to solve the insufficient sampling problem. Intuitively, a short document is *less* sufficient than a longer one, hence would need more "help" from its neighborhood. We design experiments to test this hypothesis.

Specifically, we randomly shrink each document in AP88-89 to a certain percentage of its original length. For example, a shrinkage factor of 30% means each term has 30% chance to stay, or 70% chance to be filtered out. In this way, we reduce the original data set to a new one with the same number

[1]We use the exact same data, queries, stemming and all other preprocessing techniques. The baseline results in (Liu and Croft, 2004) are confirmed.

average doc length	30%	50%	70%	100%
baseline	0.1273	0.1672	0.1916	0.2168
document expansion	0.1794	0.2137	0.2307	0.2505
optimal α	0.2	0.3	0.3	0.4
improvement(%)	41%	28%	20%	16%

Table 5: Impact on short documents (in MAP)

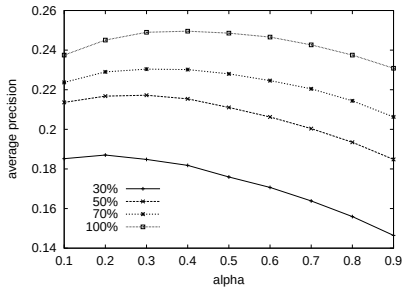


Figure 4: Performance change with respect to α

	DELM	pseudo	DELM+pseudo	Impr.(%)
AP	0.2505	0.2643	0.2726	3.14%*
LA	0.2655	0.2769	0.2901	4.77%
TREC8	0.2671	0.2716	0.2809	3.42%**
DOE	0.1898	0.1918	0.2046	6.67%***

Table 6: Combination with pseudo feedback.*,**,*** indicate that we accept the improvement hypothesis by Wilcoxon test at significant level 0.1, 0.05, 0.01 respectively.

	pseu.	inter.	combined (%)	z-score
AP	0.2643	0.2450	0.2660 (0.64%)	-0.2888
LA	0.2769	0.2662	0.2636 (-0.48%)	-1.0570
TREC8	0.2716	0.2702	0.2739 (0.84%)	-1.6938

Table 7: Performance of the interpolation algorithm combined with the pseudo feedback.

of documents but a shorter average document length.

Table 5 shows the experiment results over document sets with different average document lengths. The results indeed support our hypothesis that document expansion does help short documents more than longer ones. While we can manage to improve 41% on a 30%-length corpus, the same model only gets 16% improvement on the full length corpus.

To understand how α affects the performance we plot the sensitivity curves in Figure 4. The curves all look similar, but the optimal points slightly migrate when the average document length becomes shorter. A 100% corpus gets optimal at $\alpha = 0.4$, but 30% corpus has to use $\alpha = 0.2$ to obtain its optimum. (All optimal α values are presented in the fourth row of Table 5.)

3.4 Further improvement with pseudo feedback

Query expansion has been proved to be an effective way of utilizing corpus information to improve the query representation (Rocchio, 1971; Zhai and Lafferty, 2001a). It is thus interesting to examine whether our model can be combined with query expansion to further improve the retrieval accuracy. We use the model-based feedback proposed in (Zhai and Lafferty, 2001a) and take top 5 returned documents for feedback. There are two parameters in the model-based pseudo feedback process: the noisy parameter ρ and the interpolation parameter σ^2. We fix $\rho = 0.9$ and tune σ to optimal, and use them directly in the feedback process combined with our models. (It again means that σ is probably not optimal in our results.) The combination is conducted in the following way: (1) Retrieve documents by our DELM method; (2) Choose top 5 document to do the model-based feedback; (3) Use the expanded query model to retrieve documents again with DELM method.

Table 6 shows the experiment results (MAP); indeed, by combining DELM with pseudo feedback, we can obtain significant further improvement of performance.

As another baseline, we also tested the algorithm proposed in (Kurland and Lee, 2004). Since the algorithm overlaps with pseudo feedback process, it is not easy to further combine them. We implement its best-performing algorithm, "interpolation" (labeled as inter.), and show the results in Table 7. Here, we use the same three data sets as used in (Kurland and Lee, 2004). We tune the feedback parameters to optimal in each experiment. The second last column in Table 7 shows the performance of combination of the "interpolation" model with the pseudo feedback and its improvement percentage. The last column is the z-scores of Wilcoxon test. The negative z-scores indicate that none of the improvement is significant.

[2] (Zhai and Lafferty, 2001a) uses different notations. We change them because α has already been used in our own model.

4 Conclusions

In this paper, we proposed a novel document expansion method to enrich the document sample through exploiting the local corpus structure. Unlike previous cluster-based models, we smooth each document using a probabilistic neighborhood centered around the document itself.

Experiment results show that (1) The proposed document expansion method outperforms both the "no expansion" baselines and the cluster-based models. (2) Our model is relatively insensitive to the setting of parameter M as long as it is sufficiently large, while the parameter α should be set according to the document length; short documents need a smaller α to obtain more help from its neighborhood. (3) Document expansion can be combined with pseudo feedback to further improve performance. Since any retrieval model can be presumably applied on top of the expanded documents, we believe that the proposed technique can be potentially useful for any retrieval model.

5 Acknowledgments

This work is in part supported by the National Science Foundation under award number IIS-0347933. We thank Xiaoyong Liu for kindly providing us several processed data sets for our performance comparison. We thank Jing Jiang and Azadeh Shakery for helping improve the paper writing, and thank the anonymous reviewers for their useful comments.

References

S. F. Chen and J. Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University.

D. Hiemstra and W. Kraaij. 1998. Twenty-one at trec-7: Ad-hoc and cross-language track. In *Proc. of Seventh Text REtrieval Conference (TREC-7)*.

R. Kneser and H. Ney. 1995. Improved smoothing for m-gram languagemodeling. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*.

Oren Kurland and Lillian Lee. 2004. Corpus structure, language models, and ad hoc information retrieval. In *SIGIR '04: Proceedings of the 27th annual international conference on Research and development in information retrieval*, pages 194–201. ACM Press.

John Lafferty and Chengxiang Zhai. 2001. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of SIGIR'2001*, pages 111–119, Sept.

John Lafferty and ChengXiang Zhai. 2003. Probabilistic relevance models based on document and query generation.

Victor Lavrenko and Bruce Croft. 2001. Relevance-based language models. In *Proceedings of SIGIR'2001*, Sept.

Xiaoyong Liu and W. Bruce Croft. 2004. Cluster-based retrieval using language models. In *SIGIR '04: Proceedings of the 27th annual international conference on Research and development in information retrieval*, pages 186–193. ACM Press.

D. H. Miller, T. Leek, and R. Schwartz. 1999. A hidden markov model information retrieval system. In *Proceedings of the 1999 ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 214–221.

J. Ponte and W. B. Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of the ACM SIGIR*, pages 275–281.

J. Rocchio. 1971. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall Inc.

Amit Singhal and Fernando Pereira. 1999. Document expansion for speech retrieval. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 34–41. ACM Press.

E. Voorhees and D. Harman, editors. 2001. *Proceedings of Text REtrieval Conference (TREC1-9)*. NIST Special Publications. http://trec.nist.gov/pubs.html.

Chengxiang Zhai and John Lafferty. 2001a. Model-based feedback in the KL-divergence retrieval model. In *Tenth International Conference on Information and Knowledge Management (CIKM 2001)*, pages 403–410.

Chengxiang Zhai and John Lafferty. 2001b. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR'2001*, pages 334–342, Sept.

Towards Spoken-Document Retrieval for the Internet: Lattice Indexing For Large-Scale Web-Search Architectures

Zheng-Yu Zhou[*], **Peng Yu, Ciprian Chelba**[+], and **Frank Seide**
[*]Chinese University of Hong Kong, Shatin, Hong Kong
Microsoft Research Asia, 5F Beijing Sigma Center, 49 Zhichun Road, 100080 Beijing
[+]Microsoft Research, One Microsoft Way, Redmond WA 98052
zyzhou@se.cuhk.edu.hk, {rogeryu,chelba,fseide}@microsoft.com

Abstract

Large-scale web-search engines are generally designed for linear text. The linear text representation is suboptimal for audio search, where accuracy can be significantly improved if the search includes alternate recognition candidates, commonly represented as word lattices.

This paper proposes a method for indexing word lattices that is suitable for large-scale web-search engines, requiring only limited code changes.

The proposed method, called Time-based Merging for Indexing (TMI), first converts the word lattice to a posterior-probability representation and then merges word hypotheses with similar time boundaries to reduce the index size. Four alternative approximations are presented, which differ in index size and the strictness of the phrase-matching constraints.

Results are presented for three types of typical web audio content, podcasts, video clips, and online lectures, for phrase spotting and relevance ranking. Using TMI indexes that are only five times larger than corresponding linear-text indexes, phrase spotting was improved over searching top-1 transcripts by 25-35%, and relevance ranking by 14%, at only a small loss compared to unindexed lattice search.

1 Introduction

Search engines have become the essential tool for finding and accessing information on the Internet. The recent runaway success of podcasting has created a need for similar search capabilities to find audio on the web. As more news video clips and even TV shows are offered for on-demand viewing, and educational institutions like MIT making lectures available online, a need for audio search arises as well, because the most informative part of many videos is its dialogue.

There is still a significant gap between current web audio/video search engines and the relatively mature text search engines, as most of today's audio/video search engines rely on the surrounding text and metadata of an audio or video file, while ignoring the actual audio content. This paper is concerned with technologies for searching the audio content itself, in particular how to represent the speech content in the index.

Several approaches have been reported in the literature for indexing spoken words in audio recordings. The TREC (Text REtrieval Conference) Spoken-Document Retrieval (SDR) track has fostered research on audio-retrieval of broadcast-news clips. Most TREC benchmarking systems use broadcast-news recognizers to generate approximate transcripts, and apply text-based information retrieval to these. They achieve retrieval accuracy similar to using human reference transcripts, and ad-hoc retrieval for broadcast news is considered a "solved problem" (Garofolo, 2000). Noteworthy are the rather low word-error rates (20%) in the TREC evaluations, and that recognition errors did not lead to catastrophic failures due to redundancy of news segments and queries. However, in our scenario, unpredictable, highly variable acoustic conditions, non-native and accented speaker, informal talking style, and unlimited-domain language cause word-error rates to be much higher (40-60%). Directly searching such inaccurate speech recognition transcripts suffers from a poor recall.

A successful way for dealing with high word error rates is the use of recognition alternates (lattices) (Saraclar, 2004; Yu, 2004; Chelba, 2005). For example, (Yu, 2004) reports a 50% improvement of FOM (Figure Of Merit) for a word-spotting task in voice-mails, and (Yu, HLT2005) adopted the approach for searching personal audio collections, using a hybrid word/phoneme lattice search.

Web-search engines are complex systems involving substantial investments. For extending web search to audio search, the key problem is to find a (approximate)

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 415–422,
New York, June 2006.

representation of lattices that can be implemented in a state-of-the-art web-search engine with as little changes as possible to code and index store and without affecting its general architecture and operating characteristics.

Prior work includes (Saraclar, 2004), which proposed a direct inversion of raw lattices from the speech recognizer. No information is lost, and accuracy is the same as for directly searching the lattice. However, raw lattices contain a large number of similar entries for the same spoken word, conditioned on language-model (LM) state and phonetic cross-word context, leading to inefficient usage of storage space.

(Chelba, 2005) proposed a posterior-probability based approximate representation in which word hypotheses are merged w.r.t. word position, which is treated as a hidden variable. It easily integrates with text search engines, as the resulting index resembles a normal text index in most aspects. However, it trades redundancy w.r.t. LM state and context for uncertainty w.r.t. word position, and only achieves a small reduction of index entries. Also, time information for individual hypotheses is lost, which we consider important for navigation and previewing.

(Mangu, 2000) presented a method to align a speech lattice with its top-1 transcription, creating so-called "confusion networks" or "sausages." Sausages are a parsimonious approximation of lattices, but due to the presence of null links, they do not lend themselves naturally for matching phrases. Nevertheless, the method was a key inspiration for the present paper.

This paper is organized as follows. The next section states the requirements for our indexing method and describes the overall system architecture. Section 3 introduces our method, and Section 4 the results. Section 5 briefly describes a real prototype built using the approach.

2 Indexing Speech Lattices, Internet Scale

Substantial investments are necessary to create and operate a web search engine, in software development and optimization, infrastructure, as well as operation and maintainance processes. This poses constraints on what can practically be done when integrating speech-indexing capabilities to such an engine.

2.1 Requirements

We have identified the following special requirements for speech indexing:

- realize best possible accuracy – speech alternates must be indexed, with scores;
- provide time information for individual hits – to facilitate easy audio preview and navigation in the UI;
- encode necessary information for phrase matching – phrase matching is a basic function of a search engine and an important feature for document ranking. This is non-trivial because boundaries of recognition alternates are generally not aligned.

None of these capabilities are provided by text search engines. To add these capabilities to an existing web engine, we are facing practical constraints. First, the structure of the index store cannot be changed fundamentally. But we can reinterpret existing fields. We also assume that the index attaches a few auxiliary bits to each word hit. E.g., this is done in (early) Google (Brin, 1998) and MSN Search. These can be used for additional data that needs to be stored.

Secondly, computation and disk access should remain of similar order of magnitude as for text search. Extra CPU cycles for phrase-matching loops are possible as long as disk access remains the dominating factor. The index size cannot be excessively larger than for indexing text. This precludes direct inversion of lattices (and unfortunately also the use of phonetic lattices).

Last, while local code changes are possible, the overall architecture and dataflow cannot be changed. E.g., this forbids the use of a two-stage method as in (Yu, HLT2005).

2.2 Approach

We take a three-step approach. First, following (Chelba, 2005), we use a posterior-probability representation, as posteriors are resilient to approximations and can be quantized with only a few bits. Second, we reduce the inherent redundancy of speech lattices by merging word hypotheses with same word identity and similar time boundaries, hence the name "Time-based Merging for Indexing" (TMI). Third, the resulting hypothesis set is represented in the index by reinterpreting existing data fields and repurposing auxiliary bits.

2.3 System Architecture

Fig. 1 shows the overall architecture of a search engine for audio/video search. At indexing time, a media decoder first extracts the raw audio data from different formats of audio found on the Internet. A music detector prevents music from being indexed. The speech is then fed into a large-vocabulary continuous-speech recognizer (LVCSR), which outputs word lattices. The lattice indexer converts the lattices into the TMI representation, which is then merged into the inverted index. Available textual metadata is also indexed.

At search time, all query terms are looked up in the index. For each document containing all query terms (determined by intersection), individual hit lists of each query term are retrieved and fed into a phrase matcher to identify full and partial phrase hits. Using this information, the ranker computes relevance scores. To achieve acceptable response times, a full-scale web engine would split this process up for parallel execution on multiple servers. Finally the result presentation module will create snippets

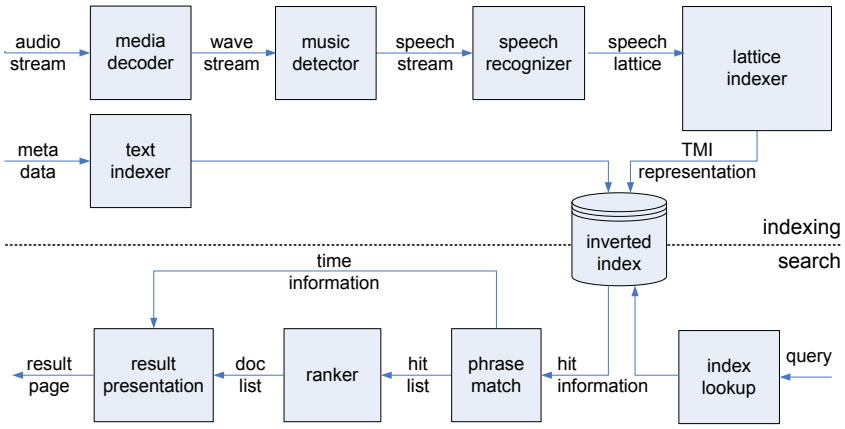


Figure 1: System Architecture.

for the returned documents and compose the result page. In audio search, snippets would contain time information for individual word hits to allow easy navigation and preview.

3 Time-based Merging for Indexing

Our previous work (Yu, IEEE2005) has shown that in a word spotting task, ranking by phrase posteriors is in theory optimal if (1) a search hit is considered relevant if the query phrase was indeed said there, and (2) the user expects a ranked list of results such that the accumulative relevance of the top-n entries of the list, averaged over a range of n, is maximized. In the following, we will first recapitulate the lattice notation and how phrase posteriors are calculated from the lattice. We then introduce time-based merging, which leads to an approximate representation of the original lattice. We will describe two strategies of merging, one by directly clustering word hypotheses (arc-based merging) and one by grouping lattice nodes (node-based merging).

3.1 Posterior Lattice Representation

A lattice $\mathcal{L} = (\mathcal{N}, \mathcal{A}, n_{\text{start}}, n_{\text{end}})$ is a directed acyclic graph (DAG) with $\mathcal{N}$ being the set of nodes, $\mathcal{A}$ is the set of arcs, and n_{start}, $n_{\text{end}} \in \mathcal{N}$ being the unique initial and unique final node, respectively. Nodes represent times and possibly context conditions, while arcs represent word or phoneme hypotheses.[1]

Each node $n \in \mathcal{N}$ has an associated time $t[n]$ and possibly an acoustic or language-model context condition. Arcs are 4-tuples $a = (S[a], E[a], I[a], w[a])$. $S[a]$, $E[a] \in \mathcal{N}$ denote the start and end node of the arc. $I[a]$ is the word identity. Last, $w[a]$ shall be a weight assigned to the arc by the recognizer. Specifically, $w[a] = p_{\text{ac}}(a)^{1/\lambda} \cdot P_{\text{LM}}(a)$ with acoustic likelihood $p_{\text{ac}}(a)$, LM probability P_{LM}, and LM weight λ.

[1] Alternative definitions of lattices are possible, e.g. nodes representing words and arcs representing word transitions.

In addition, we define *paths* $\pi = (a_1, \cdots, a_K)$ as *sequences* of connected arcs. We use the symbols S, E, I, and w for paths as well to represent the respective properties for entire paths, i.e. the path start node $S[\pi] = S[a_1]$, path end node $E[\pi] = E[a_K]$, path label sequence $I[\pi] = (I[a_1], \cdots, I[a_K])$, and total path weight $w[\pi] = \prod_{k=1}^{K} w[a_k]$.

Based on this, we define *arc posteriors* $P_{\text{arc}}[a]$ and *node posteriors* $P_{\text{node}}[n]$ as

$$P_{\text{arc}}[a] = \frac{\alpha_{S[a]} \cdot w[a] \cdot \beta_{E[a]}}{\alpha_{n_{\text{end}}}} \quad ; \quad P_{\text{node}}[n] = \frac{\alpha_n \cdot \beta_n}{\alpha_{n_{\text{end}}}},$$

with *forward-backward probabilities* α_n, β_n defined as:

$$\alpha_n = \sum_{\pi: S[\pi]=n_{\text{start}} \wedge E[\pi]=n} w[\pi] \quad ; \quad \beta_n = \sum_{\pi: S[\pi]=n \wedge E[\pi]=n_{\text{end}}} w[\pi]$$

α_n and β_n can be conveniently computed using the well-known forward-backward recursion, e.g. (Wessel, 2000).

With this, an alternative equivalent representation is possible by using word posteriors as arc weights. The *posterior lattice* representation stores four fields with each edge: $S[a]$, $E[a]$, $I[a]$, and $P_{\text{arc}}[a]$, and two fields with each node: $t[n]$, and $P_{\text{node}}[a]$.

With the posterior lattice representation, the phrase posterior of query string Q is computed as

$$P(*, t_s, Q, t_e, *|O) = \sum_{\substack{\pi=(a_1,\cdots,a_K): \\ t[S[\pi]]=t_s \wedge t[E[\pi]]=t_e \wedge I[\pi]=Q}} \frac{P_{\text{arc}}[a_1] \cdots P_{\text{arc}}[a_K]}{P_{\text{node}}[S[a_2]] \cdots P_{\text{node}}[S[a_K]]}. \quad (1)$$

This posterior representation is lossless. Its advantage is that posteriors are much more resiliant to approximations than acoustic likelihoods. This paves the way for lossy approximations aiming at reducing lattice size.

3.2 Time-based Merging for Indexing

First, (Yu, HLT2005) has shown that node posteriors can be replaced by a constant, with no negative effect on

search accuracy. This approximation simplifies the denominator in Eq. 1 to p_{node}^{K-1}.

We now merge all nodes associated with the same time points. As a result, the connection condition for two arcs depends only on the boundary time point. This operation gave the name Time-based Merging for Indexing.

TMI stores arcs with start and end time, while discarding the original node information that encoded dependency on LM state and phonetic context. This form is used, e.g., by (Wessel, 2000). Lattices are viewed as sets of *items* $h = (ts[h], dur[h], I[h], P[h])$, with $ts[h]$ being the start time, $dur[h]$ the time duration, $I[h]$ the word identity, and $P[h]$ the posterior probability. Arcs with same word identity and time boundaries but different start/end nodes are merged together, their posteriors being summed up.

These item sets can be organized in an inverted index, similar to a text index, for efficient search. A text search engine stores at least two fields with each word hit: word position and document identity. For TMI, two more fields need to be stored: duration and posterior. Start times can be stored by repurposing the word-position information. Posterior and duration go into auxiliary bits. If the index has the ability to store side information for documents, bits can be saved in the main index by recording all time points in a look-up table, and storing start times and durations as table indices instead of absolute times. This works because the actual time values are only needed for result presentation. Note that the TMI index is really an extension of a linear-text index, and the same code base can easily accomodate indexing both speech content and textual metadata.

With this, multi-word phrase matches are defined as a sequence of items $h_1 ... h_K$ matching the query string $(Q = (I[h_1], \cdots, I[h_K]))$ with matching boundaries $(ts[h_i] + dur[h_i] = ts[h_{i+1}])$. The phrase posterior is calculated (using the approximate denominator) as

$$P(*, t_s, Q, t_e, *|O) \approx \sum \frac{P[h_1] \cdots P[h_K]}{p_{\text{node}}^{K-1}}, \quad (2)$$

summing over all item sequences with $t_s = ts[h_1]$ and $t_e = ts[h_K] + dur[h_K]$.

Regular text search engines can not directly support this, but the code modification and additional CPU cost is small. The major factor is disk access, which is still linear with the index size.

We call this index representation "TMI-base." It provides a substantial reduction of number of index entries compared to the original lattices. However, it is obviously an approximative representation. In particular, there are now conditions under which two word hypotheses can be matched as part of a phrase that were not connected in the original lattice. This approximation seems sensible, though, as the words involved are still required to have precisely matching word boundaries. In fact it has been shown that this representation can be used for direct word-error minimization during decoding (Wessel, 2000).

Table 1: Test corpus summary.

test set	duration	#segments	#keywords (#multi-word)	WER [%]
podcasts	1.5h	367	3223 (1709)	45.8
videos	1.3h	341	2611 (1308)	50.8
lectures	169.6h	66102	96 (74)	54.8

For further reduction of the index size, we are now relaxing the merging condition. The next two sections will introduce two alternate ways of merging.

3.3 Arc-Based Merging

A straightforward way is to allow tolerance of time boundaries. Practically, this is done by the following bottom-up clustering procedure:

- collect arcs with same word identity;
- find the arc a^* with the best posterior, set the resulting item time boundary same as a^*;
- merge all overlapping arcs a satisfying $t[S[a^*]] - \triangle_1 \leq t[S[a]] \leq t[S[a^*]] + \triangle_1$ and $t[E[a^*]] - \triangle_1 \leq t[E[a]] \leq t[E[a^*]] + \triangle_1$;
- repeat with remaining arcs.

We call this method "TMI-arc" to denote its origin from direct clustering of arcs.

Note that the resulting structure can generally not be directly represented as a lattice anymore, as formally connected hypotheses now may have slightly mismatching time boundaries. To compensate for this, the item connection condition in phrase matching needs to be relaxed as well: $ts[h_{i+1}] - \triangle_1 \leq ts[h_i] + dur[h_i] \leq ts[h_{i+1}] + \triangle_1$.

The storage cost for each TMI-arc item is same as for TMI-base, while the *number* of items will be reduced.

3.4 Node-Based Merging

An alternative way is to group ranges of time points, and then merge hypotheses whose time boundaries got grouped together.

The simplest possibility is to quantize time points into fixed intervals, such as 250 ms. Hypotheses are merged if their quantized time boundaries are identical. This method we call "TMI-timequant."

Besides reducing index size by allowing more item merging, TMI-timequant has another important property: since start times and duration are heavily quantized, the number of bits used for storing the information with the items in the index can be significantly reduced.

The disadvantage of this method is that loops are frequently being generated this way (quantized duration of 0), providing sub-optimal phrase matching constraints.

To alleviate for this problem, we modify the merging by forbidding loops to be created: Two time points can be

Table 2: Lattice search accuracy on different dataset.

setup	best path			raw lattice		
keywords	all	sing.	mult.	all	sing.	mult.
Phrase spotting, FOM[%]						
podcasts	55.0	59.9	50.1	69.5	74.7	64.2
videos	47.0	50.6	43.0	64.4	67.4	61.1
lectures	65.5	69.5	47.1	77.0	80.8	58.8
Relevance ranking, mAP[%]						
lectures	52.6	52.7	52.6	61.6	66.4	60.2

grouped together if (1) their difference is below a threshold (like 250 ms); and (2) if there is no word hypothesis starting and ending in the same group. As a refinement, the second point is relaxed by a pruning threshold in that hypotheses with posteriors below the threshold will not block nodes merging.

Amongst the manifold of groupings that satisfy these two conditions, the one leading to the smallest number of groups is considered the optimal solution. It can be found using dynamic programming:

- line up all existing time boundaries in ascending order, $t_i < t_{i+1}, i = 1, \cdots, N$;
- for each time point t_i, find out the furthest time point that it can be grouped with given the constraints, denoting its index as $T[t_i]$;
- set group count $C[t_0] = 1$; $C[t_i] = \infty$, $i > 0$;
- set backpointer $B[t_0] = -1$; $B[t_i] = t_i$, $i > 0$;
- for $i = 1, \cdots, N$:
  - for $j = i+1, \cdots, T[t_i]$: if $C[t_{j+1}] > C[t_i]+1$:
    - $C[t_{j+1}] = C[t_i] + 1$;
    - $B[t_{j+1}] = t_i$;
- trace back and merge nodes:
  - set $k = N$, repeat until $k = -1$:
    - group time points from $B[t_k]$ to t_{k-1};
    - $k = B[t_k]$.

This method can be applied to the TMI-base representation, or alternatively directly to the posterior lattice. In this case, the above algorithm needs to be adapted to operate on nodes rather than time points. The above method is called "TMI-node."

If, as mentioned before, times and durations are stored as indexes into a look-up table, TMI-node is highly space efficient. In most cases, the index difference between end and start point is 1, and in practical terms, the index difference can be capped by a small number below 10.

4 Results

4.1 Setup

We have evaluated our system on three different corpora, in an attempt to represent popular types of audio currently found on the Internet:

- podcasts: short clips ranging from mainstream media like ABC and CNN to non-professionally produced edge content;
- video clips, acquired from MSN Video;
- online lectures: a subset of the MIT iCampus lecture collection (Glass, 2004).

In relation to our goal of web-scale indexing, the podcast and video sets are miniscule in size (about 1.5 hours each). Nevertheless they are suitable for investigating the effectiveness of the TMI method w.r.t. phrase spotting accuracy. Experiments on relevance ranking were conducted only on the much larger lecture set (170 hours).

For the iCampus lecture corpus, the same set of queries was used as in (Chelba, 2005), which was collected from a group of users. Example keywords are *computer science* and *context free grammar*. On the other two sets, an automatic procedure described in (Seide, 2004) was used to select keywords. Example keywords are *playoffs*, *beach Florida*, and *American Express financial services.*

A standard speaker-independent trigram LVCSR system was used to generate raw speech lattices. For video and podcasts, models were trained on a combination of telephone conversations (Switchboard), broadcast news, and meetings, downsampled to 8 kHz, to accomodate for a wide range of audio types and speaking styles. For lectures, an older setup was used, based on a dictation engine without adaptation to the lecture task. Due to the larger corpus size, lattices for lectures were pruned much more sharply. Word error rates (WER) and corpus setups are listed in Table 1. It should be noted that the word-error rates vary greatly within the podcast and video corpora, ranging from 30% (clean broadcast news) to over 80% (accented reverberated speech with a cheering crowd).

Each indexing method is evaluated by a phrase spotting task and a document retrieval task.

4.1.1 Phrase Spotting

We use the "Figure Of Merit" (FOM) metric defined by NIST for word-spotting evaluations. In its original form, FOM is the detection/false-alarm curve averaged over the range of [0..10] false alarms per hour per keyword. We generalized this metric to spotting of phrases, which can be multi-word or single-word. A multi-word phrase is matched if all of its words match in order.

Since automatic word alignment can be troublesome for long audio files in the presence of errors in the reference transcript, we reduced the time resolution of the FOM metric and used the sentence as the basic time unit. A phrase hit is considered correct if an actual occurence of the phrase is found in the same sentence. Multiple hits of the same phrase within one sentence are counted as a single hit, their posterior probabilities being summed up for ranking.

The segmentation of the audio files is based on the reference transcript. Segments are on average about 10 seconds long. In a real system, sentence boundaries are of course unknown, but previous experiments have shown

Table 3: Comparison of different indexing methods. Only results for multi-words queries are shown, because results for single-word queries are identical across lattice-indexing methods (approximately identical in the case of pruning.)

dataset	podcasts		videos		lectures		
	FOM [%]	size	FOM [%]	size	FOM [%]	mAP [%]	size
bestpath	50.1	1.1	43.0	1.0	47.1	52.6	1.0
raw lattice	64.2	527.6	61.1	881.7	58.8	60.2	23.3
$P_{\text{node}} = \text{const}$	64.3	527.6	61.1	881.7	58.8	60.3	23.3
no pruning							
TMI-base	65.3	55.2	62.6	78.8	58.8	60.2	7.7
TMI-arc	62.9	16.1	58.5	20.7	57.9	60.1	4.4
TMI-timequant	66.7	15.4	64.2	19.5	58.8	60.3	4.5
TMI-node	66.5	20.7	63.4	27.6	58.7	59.7	4.4
PSPL	68.9	182.0	66.2	212.0	58.7	61.0	21.2
pruned to about 5 entries per spoken word							
TMI-base	62.1	5.6	54.1	5.1	57.0	60.3	4.5
TMI-arc	60.7	4.6	53.6	5.0	57.9	60.1	4.4
TMI-timequant	63.1	4.7	57.1	5.1	58.8	60.3	4.5
TMI-node	63.7	4.6	57.7	5.1	58.7	59.7	4.4
PSPL	57.3	6.0	49.8	5.8	53.6	61.0	4.4

that the actual segmentation does not have significant impact on the results.

4.1.2 Relevance Ranking

The choice and optimization of a relevance ranking formula is a difficult problem that is beyond the scope of this paper. We chose a simple document ranking method as described in (Chelba, 2005):

Given query $Q = (q_1, \cdots, q_L)$, for each document D, expected term frequencies (ETF) of all sub-strings $Q_{[i,j]} = (q_i, \cdots, q_j)$ are calculated:

$$\text{ETF}(Q_{[i,j]}|D) = \sum_{t_s,t_e} P(*, t_s, Q_{[i,j]}, t_e, *|O, D) \quad (3)$$

A document is returned if all query words are present. The relevance score is calculated as

$$S(D, Q) = \sum_{i=1}^{L} \sum_{j=i}^{L} w_{j-i} \log[1 + \text{ETF}(Q_{[i,j]}|D)] \quad (4)$$

where the weights w_ℓ have the purpose to give higher weight to longer sub-strings. They were chosen as $w_\ell = 1 + 1000 \cdot \ell$, no further optimization was performed.

Only the lecture set is used for document retrieval evaluation. The whole set consists of 169 documents, with an average of 391 segments in each document. The evaluation metric is the mean average precision (mAP) as computed by the standard `trec_eval` package used by the TREC evaluations (NIST, 2005). Since actual relevance judgements were not available for this corpus, we use the output of a state-of-the-art text retrieval engine on the ground truth transcripts as the reference. The idea is that if human judgements are not available, the next best thing to do is to assess how close our spoken-document retrieval system gets to a text engine applied to reference transcripts. Although one should take the absolute mAP scores with a pinch of salt, we believe that comparing the relative changes of these mAP scores is meaningful.

4.2 Lattice Search and Best Path Baseline

Table 2 lists the word spotting and document retrieval result of direct search in the original raw lattice, as well as for searching the top-1 path. Results are listed separately for single- and multi-word queries. For the phrase-spotting task, a consistent about 15% improvement is observed on all sets, re-emphasizing the importance of searching alternates. For document retrieval, the accuracy (mAP) is also significantly improved from 53% to 62%.

4.2.1 Comparing Indexing Methods

Table 3 compares different indexing methods with respect to search accuracy and index size. We only show results for multi-words queries results, as it can be shown that results for single-word queries must be identical. The index size is measured as index entries per spoken word, i.e. it does not reflect that different indexing methods may require different numbers of bits in the actual index store.

In addition to four types of TMI methods, we include an alternative posterior-lattice indexing method in our comparison called PSPL (position-specific posterior lattices) (Chelba, 2005). A PSPL index is constructed by enumerating all paths through a lattice, representing each path as a linear text, and adding each text to the index, each time starting over from word position 1. Each word hypothesis on each path is assigned the posterior probability of the entire path. Instances of the same word occuring at the same text position are merged, accumulating their posterior probabilities. This way, each index entry represents the posterior probability that a word occurs at a particular position in the actual spoken word sequence. PSPL is an attractive alternative to the work presented in

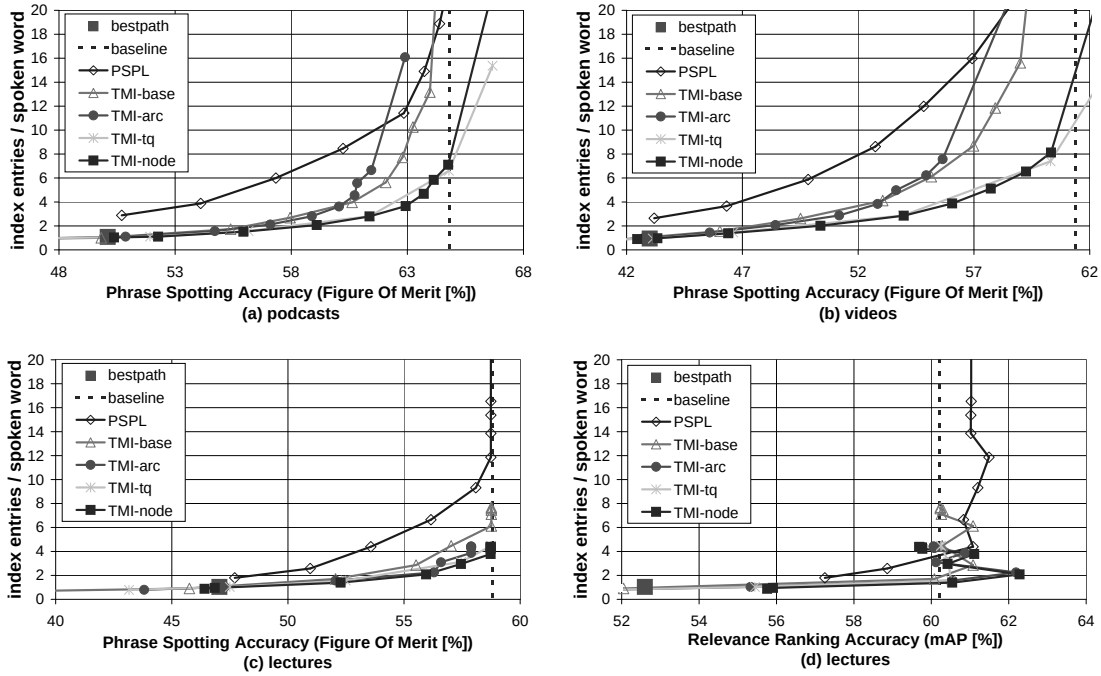


Figure 2: Index size vs. accuracy for different pruning thresholds for word-spotting on (a) podcasts, (b) videos, (c) lectures, and (d) relevance ranking for lectures.

this paper because it continues to use the notion of a word position instead of time, with the advantage that existing implementations of phrase-matching conditions apply without modification.

The results show that, comparing with the direct raw-lattice search, all indexing methods have only slight impact on both word spotting and document retrieval accuracies. Against our expectation, in many cases *improved* accuracies are observed. These are caused by creating additonal paths compared to the original lattice, improving recall. It is not yet clear how to exploit this in a systematic manner.

W.r.t. storage efficiency, the TMI merging methods have about 5 times less index entries than the original lattice for lectures (and an order of magnitude less for podcasts and videos that were recognized with rather wasteful pruning thresholds). This can be further improved by pruning.

4.2.2 Pruning

Index size and accuracy can be balanced by pruning low-scoring index entries. Experiments have shown that the optimal pruning strategy differs slightly from method to method. For the TMI set, the index is pruned by removing all entries with posterior probabilities below a certain fixed threshold. In addition, for TMI-node we enforce that the best path is not pruned. For PSPL, an index entry at a particular word position is removed if its posterior is worse by a fixed factor compared to the best index entry for the *same* word position. This also guarantees that the best path is never pruned.

Fig. 2 depicts the trade-off of size and accuracy for different indexing methods. TMI-node provides the best trade-off. The last block of Table 3 shows results for all indexing methods when pruned with the respective pruning thresholds adjusted such that the number of index entries is approximately five times that for the top-1 transcript. We chose this size because reducing the index size still has limited impact on accuracy (0.5-points for podcasts, 3.5 for videos, and none for lectures) while keeping operating characteristics (storage size, CPU, disk) within an order of magnitude from text search.

5 The System

The presented technique was implemented in a research prototype shown in Fig. 3. About 780 hours of audio documents, including video clips from MSN Video and audio files from most popular podcasts, were indexed. The index is disk-based, its size is 830 MB, using a somewhat wasteful XML representation for research convenience. Typically, searches are executed within 0.5 seconds.

The user interface resembles a typical text search engine. A media player is embedded for immediate within-page playback. Snippets are generated for previewing the search results. Each word in a snippet has its original time point associated, and a click on it positions the media player to the corresponding time in the document.

6 Conclusion

We targeted the paper to the task of searching audio content from the Internet. Aiming at maximizing reuse of existing web-search engines, we investigated how best to

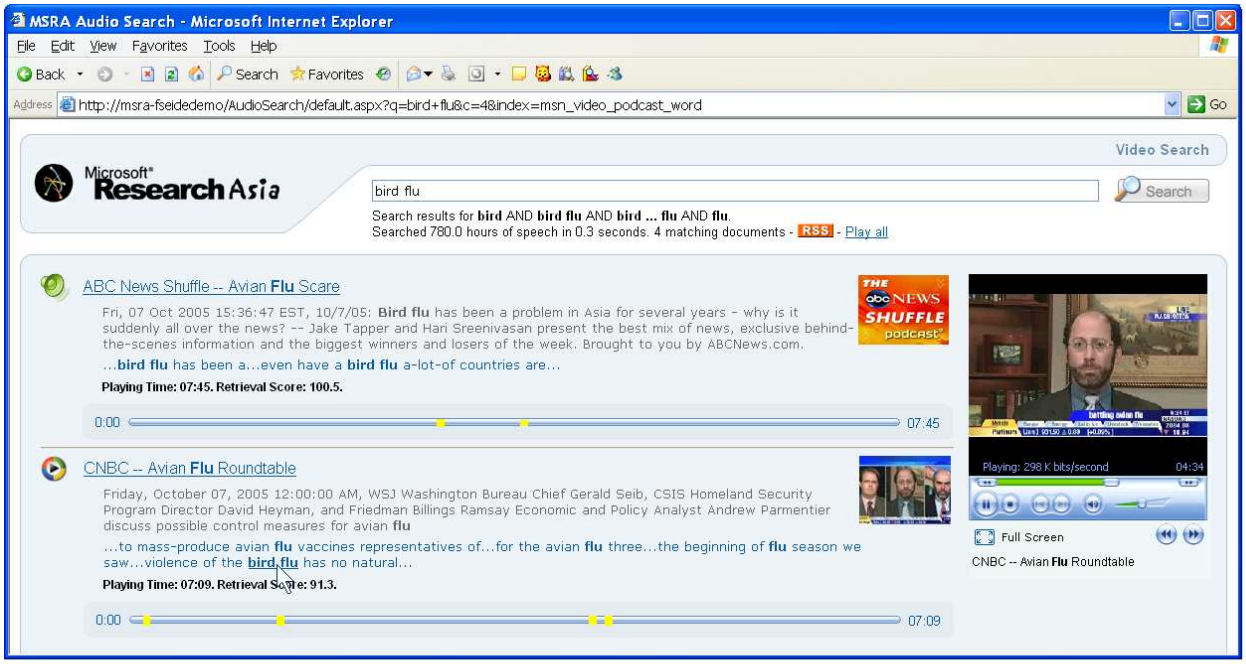


Figure 3: Screenshot of the video/audio-search prototype. For each document, in addition to the title and description text from meta-data, the system displays recognition-transcript snippets around the audio hits, e.g. "... **bird flu** has been a ..." in the first document. Clicking on a word in a snippet starts playing back the video at that position using the embedded video player.

represent important lattice properties – recognition alternates with scores, time boundaries, and phrase-matching constraints – in a form suitable for large-scale web-search engines, while requiring only limited code changes.

The proposed method, Time-based Merging for Indexing (TMI), first converts the word lattice to a posterior-probability representation and then merges word hypotheses with similar time boundaries to reduce the index size. Four approximations were presented, which differ in size and the strictness of phrase-matching constraints.

Results were presented for three typical types of web audio content – podcasts, video clips, and online lectures – for phrase spotting and relevance ranking. Using TMI indexes that are only five times larger than corresponding linear-text indexes, accuracy was improved over searching top-1 transcripts by 25-35% for word spotting and 14% for relevance ranking, very close to what is gained by a direct search of unindexed lattices.

Practical feasibility has been demonstrated by a research prototype with 780 hours indexed audio, which completes searches within 0.5 seconds.

To our knowledge, this is also the first paper to report speech recognition results for podcasts.

7 Acknowledgements

The authors wish to thank Jim Glass and T. J. Hazen at MIT for providing the iCampus data.

References

S. Brin and L. Page, The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems, 30(1-7):107-117*.

C. Chelba and A. Acero, Position specific posterior lattices for indexing speech. *Proc. ACL'2005*, Ann Arbor, 2005.

J. Garofolo, TREC-9 Spoken Document Retrieval Track. National Institute of Standards and Technology, `http://trec.nist.gov/pubs/trec9/sdrt9_slides/sld001.htm`.

J. Glass, T. J. Hazen, L. Hetherington, C. Wang, Analysis and Processing of Lecture Audio data: Preliminary investigation. *Proc. HLT-NAACL'2004 Workshop: Interdisciplinary Approaches to Speech Indexing and Retrieval*, Boston, 2004.

L. Mangu, E. Brill, A. Stolcke, Finding Consensus in Speech Recognition: Word Error Minimization and Other Applications of Confusion Networks. *Computer, Speech and Language, 14(4):373-400*.

MSN Video. `http://video.msn.com`.

The TREC evaluation package. `http://www-lpir.nist.gov/projects/trecvid/trecvid.tools/trec_eval`.

M. Saraclar, R. Sproat, Lattice-based search for spoken utterance retrieval. *Proc. HLT'2004*, Boston, 2004.

F. Seide, P. Yu, *et al.*, Vocabulary-independent search in spontaneous speech. *Proc. ICASSP'2004*, Montreal, 2004.

F. Wessel, R. Schlüter, and H. Ney, Using posterior word probabilities for improved speech recognition. *Proc. ICASSP'2000*, Istanbul, 2000.

P. Yu, K. J. .Chen, L. Lu, F. Seide, Searching the Audio Notebook: Keyword Search in Recorded Conversations. *Proc. HLT'2005*, Vancouver, 2005.

P. Yu, K. J. Chen, C. Y. Ma, F. Seide, Vocabulary-Independent Indexing of Spontaneous Speech, IEEE transaction on Speech and Audio Processing, Vol.13, No.5, Special Issue on Data Mining of Speech, Audio and Dialog.

P. Yu, F. Seide, A hybrid word / phoneme-based approach for improved vocabulary-independent search in spontaneous speech. *Proc. ICLSP'04*, Jeju, 2004.

A fast finite-state relaxation method for enforcing global constraints on sequence decoding

Roy W. Tromble and **Jason Eisner**
Department of Computer Science and Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD 21218
{royt,jason}@cs.jhu.edu

Abstract

We describe finite-state constraint relaxation, a method for applying global constraints, expressed as automata, to sequence model decoding. We present algorithms for both hard constraints and binary soft constraints. On the CoNLL-2004 semantic role labeling task, we report a speedup of at least 16x over a previous method that used integer linear programming.

1 Introduction

Many tasks in natural language processing involve sequence labeling. If one models long-distance or global properties of labeled sequences, it can become intractable to find ("decode") the best labeling of an unlabeled sequence.

Nonetheless, such global properties can improve the accuracy of a model, so recent NLP papers have considered practical techniques for decoding with them. Such techniques include Gibbs sampling (Finkel et al., 2005), a general-purpose Monte Carlo method, and integer linear programming (ILP), (Roth and Yih, 2005), a general-purpose exact framework for NP-complete problems.

Under generative models such as hidden Markov models, the probability of a labeled sequence depends only on its local properties. The situation improves with discriminatively trained models, such as conditional random fields (Lafferty et al., 2001), which do efficiently allow features that are functions of the entire *observation* sequence. However, these features can still only look locally at the *label* sequence. That is a significant shortcoming, because in many domains, hard or soft global constraints on the label sequence are motivated by common sense:

- For named entity recognition, a phrase that appears multiple times should tend to get the same label each time (Finkel et al., 2005).
- In bibliography entries (Peng and McCallum, 2004), a given field (author, title, etc.) should be filled by at most one substring of the input, and there are strong preferences on the co-occurrence and order of certain fields.
- In seminar announcements, a given field (speaker, start time, etc.) should appear with at most one value in each announcement, although the field and value may be repeated (Finkel et al., 2005).
- For semantic role labeling, each argument should be instantiated only once for a given verb. There are several other constraints that we will describe later (Roth and Yih, 2005).

A popular approximate technique is to hypothesize a list of possible answers by decoding without any global constraints, and then *rerank* (or prune) this n-best list using the full model with all constraints. Reranking relies on the local model being "good enough" that the globally best answer appears in its n-best list. Otherwise, reranking can't find it.

In this paper, we propose "constraint relaxation," a simple exact alternative to reranking. As in reranking, we start with a weighted lattice of hypotheses proposed by the local model. But rather than restrict to the n best of these according to the local model, we aim to directly extract the *one* best according to the *global* model. As in reranking, we hope that the local constraints alone will work well, but if they do not, the penalty is not incorrect decoding, but longer runtime as we gradually fold the global constraints into the lattice. Constraint relaxation can be used whenever the global constraints can be expressed as regular languages over the label sequence.

In the worst case, our runtime may be exponential in the number of constraints, since we are considering an intractable class of problems. However, we show that in practice, the method is quite effective at rapid decoding under global hard constraints.

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 423–430,
New York, June 2006.

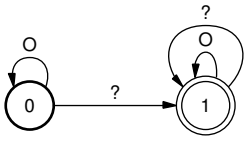


Figure 1: An automaton expressing the constraint that the label sequence cannot be $\texttt{O}^*$. Here ? matches any symbol except O.

The remainder of the paper is organized as follows: In §2 we describe how finite-state automata can be used to apply global constraints. We then give a brute-force decoding algorithm (§3). In §4, we present a more efficient algorithm for the case of hard constraints. We report results for the semantic role labeling task in §5. §6 treats soft constraints.

2 Finite-state constraints

Previous approaches to global sequence labeling—Gibbs sampling, ILP, and reranking—seem motivated by the idea that standard sequence methods are incapable of considering global constraints at all.

In fact, finite-state automata (FSAs) are powerful enough to express many long-distance constraints. Since all finite languages are regular, *any* constraint over label sequences of bounded length is finite-state. FSAs are more powerful than n-gram models. For example, the regular expression $\Sigma^*\texttt{X}\Sigma^*\texttt{Y}\Sigma^*$ matches only sequences of labels that contain an X before a Y. Similarly, the regular expression $\neg(\texttt{O}^*)$ requires at least one non-O label; it compiles into the FSA of Figure 1.

Note that this FSA is in one or the other of its two states according to whether it has encountered a non-O label yet. In general, the current state of an FSA records properties of the label sequence prefix read so far. The FSA needs enough states to keep track of whether the label sequence as a whole satisfies the global constraint in question.

FSAs are a flexible approach to constraints because they are closed under logical operations such as disjunction (union) and conjunction (intersection). They may be specified by regular expressions (Karttunen et al., 1996), in a logical language (Vaillette, 2004), or directly as FSAs. They may also be weighted to express soft constraints.

Formally, we pose the decoding problem in terms of an observation sequence $\mathbf{x} \in \mathcal{X}^*$ and possible label sequences $\mathbf{y} \in \mathcal{Y}^*$. In many NLP tasks, $\mathcal{X}$ is the set of words, and $\mathcal{Y}$ the tags. A lattice L: $\mathcal{Y}^* \mapsto \mathbb{R}$ maps label sequences to weights, and is encoded as a weighted FSA. Constraints are formally the same—any function C: $\mathcal{Y}^* \mapsto \mathbb{R}$ is a constraint, including weighted features from a classifier or probabilistic model. In this paper we will consider only constraints that are weighted in particular ways.

Given a lattice L and constraints $\mathcal{C}$, we seek

$$\mathbf{y}^* \stackrel{\text{def}}{=} \operatorname*{argmax}_{\mathbf{y}} \left(L(\mathbf{y}) + \sum_{C \in \mathcal{C}} C(\mathbf{y}) \right). \quad (1)$$

We assume the lattice L is generated by a model M: $\mathcal{X}^* \mapsto (\mathcal{Y}^* \mapsto \mathbb{R})$. For a given observation sequence $\mathbf{x}$, we put $L = M(\mathbf{x})$. One possible model is a finite-state transducer, where $M(\mathbf{x})$ is an FSA found by composing the transducer with $\mathbf{x}$. Another is a CRF, where $M(\mathbf{x})$ is a lattice with sums of log-potentials for arc weights.[1]

3 A brute-force finite-state decoder

To find the best constrained labeling in a lattice, $\mathbf{y}^*$, according to (1), we could simply intersect the lattice with all the constraints, then extract the best path.

Weighted FSA intersection is a generalization of ordinary unweighted FSA intersection (Mohri et al., 1996). It is customary in NLP to use the so-called tropical semiring, where weights are represented by their natural logarithms and summed rather than multiplied. Then the intersected automaton $L \cap C$ computes

$$(L \cap C)(\mathbf{y}) \stackrel{\text{def}}{=} L(\mathbf{y}) + C(\mathbf{y}) \quad (2)$$

To find $\mathbf{y}^*$, one would extract the best path in $L \cap C_1 \cap C_2 \cap \cdots$ using the Viterbi algorithm, or Dijkstra's algorithm if the lattice is cyclic. This step is fast if the intersected automaton is small.

The problem is that the multiple intersections in $L \cap C_1 \cap C_2 \cap \cdots$ can quickly lead to an FSA with an intractable number of states. The intersection of two finite-state automata produces an automaton

[1] For example, if M is a simple linear-chain CRF, $L(\mathbf{y}) = \sum_{i=1}^{n} f(y_{i-1}, y_i) + g(x_i, y_i)$. We build $L = M(\mathbf{x})$ as an acyclic FSA whose state set is $\mathcal{Y} \times \{1, 2, \ldots n\}$, with transitions $(y', i-1) \rightarrow (y, i)$ of weight $f(y', y) + g(x_i, y)$.

with the cross product state set. That is, if F has m states and G has n states, then $F \cap G$ has up to mn states (fewer if some of the mn possible states do not lie on any accepting path).

Intersection of many such constraints, even if they have only a few states each, quickly leads to a combinatorial explosion. In the worst case, the size, in states, of the resulting lattice is exponential in the number of constraints. To deal with this, we present a constraint relaxation algorithm.

4 Hard constraints

The simplest kind of constraint is the hard constraint. Hard constraints are necessarily binary—either the labeling satisfies the constraint, or it violates it. Violation is fatal—the labeling produced by decoding must satisfy each hard constraint.

Formally, a hard constraint is a mapping C: $\mathcal{Y}^* \mapsto \{0, -\infty\}$, encoded as an unweighted FSA. If a string satisfies the constraint, recognition of the string will lead to an accepting state. If it violates the constraint, recognition will end in a non-accepting state.

Here we give an algorithm for decoding with a set of such constraints. Later (§6), we discuss the case of binary soft constraints. In what follows, we will assume that there is always at least one path in the lattice that satisfies all of the constraints.

4.1 Decoding by constraint relaxation

Our decoding algorithm first relaxes the global constraints and solves a simpler problem. In particular, we find the best labeling according to the model,

$$\mathbf{y}_0^* \stackrel{\text{def}}{=} \operatorname*{argmax}_{\mathbf{y}} L(\mathbf{y}) \tag{3}$$

ignoring *all* the constraints in $\mathcal{C}$.

Next, we check whether $\mathbf{y}_0^*$ satisifies the constraints. If so, then we are done—$\mathbf{y}_0^*$ is also $\mathbf{y}^*$. If not, then we reintroduce the constraints. However, rather than include all at once, we introduce them only as they are violated by successive solutions to the relaxed problems: $\mathbf{y}_0^*$, $\mathbf{y}_1^*$, etc. We define

$$\mathbf{y}_1^* \stackrel{\text{def}}{=} \operatorname*{argmax}_{\mathbf{y}} \left(L(\mathbf{y}) + C(\mathbf{y})\right) \tag{4}$$

for some constraint C that $\mathbf{y}_0^*$ violates. Similarly, $\mathbf{y}_2^*$ satisfies an additional constraint that $\mathbf{y}_1^*$ violates,

HARD-CONSTRAIN-LATTICE($L, \mathcal{C}$):
1. $\mathbf{y} :=$ Best-Path(L)
2. **while** $\exists C \in \mathcal{C}$ such that $C(\mathbf{y}) = -\infty$:
3. $\quad L := L \cap C$
4. $\quad \mathcal{C} := \mathcal{C} - \{C\}$
5. $\quad \mathbf{y} :=$ Best-Path(L)
6. **return** $\mathbf{y}$

Figure 2: Hard constraints decoding algorithm.

and so on. Eventually, we find some k for which $\mathbf{y}_k^*$ satisfies all constraints, and this path is returned.

To determine whether a labeling $\mathbf{y}$ satisfies a constraint C, we represent $\mathbf{y}$ as a straight-line automaton and intersect with C, checking the result for non-emptiness. This is equivalent to string recognition.

Our hope is that, although intractable in the worst case, the constraint relaxation algorithm will operate efficiently in practice. The success of traditional sequence models on NLP tasks suggests that, for natural language, much of the correct analysis can be recovered from local features and constraints alone. We suspect that, as a result, global constraints will often be easy to satisfy.

Pseudocode for the algorithm appears in Figure 2. Note that line 2 does *not* specify how to choose C from among multiple violated constraints. This is discussed in §7. Our algorithm resembles the method of Koskenniemi (1990) and later work. The difference is that there lattices are unweighted and may not contain a path that satisfies all constraints, so that the order of constraint intersection matters.

5 Semantic role labeling

The semantic role labeling task (Carreras and Màrques, 2004) involves choosing instantiations of verb arguments from a sentence for a given verb. The verb and its arguments form a *proposition*. We use data from the CoNLL-2004 shared task—the PropBank (Palmer et al., 2005) annotations of the Penn Treebank (Marcus et al., 1993), with sections 15–18 as the training set and section 20 as the development set. Unless otherwise specified, all measurements are made on the development set.

We follow Roth and Yih (2005) exactly, in order to compare system runtimes. They, in turn, follow Hacioglu et al. (2004) and others in labeling only the heads of syntactic chunks rather than all words. We label only the core arguments (`A0`–`A5`), treating

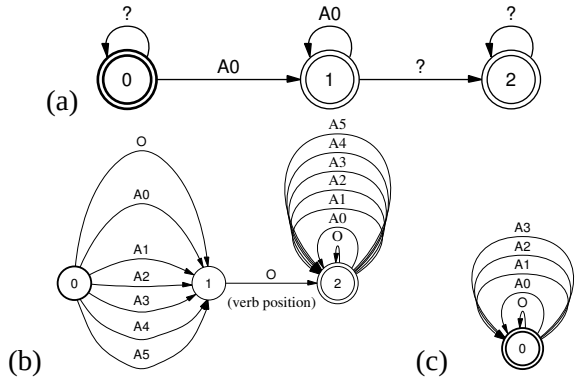


Figure 4: Automata expressing NO DUPLICATE A0 (? matches anything but A0), KNOWN VERB POSITION[2], and DISALLOW ARGUMENTS[A4,A5].

adjuncts and references as O.

Figure 3 shows an example sentence from the shared task. It is marked with an IOB phrase chunking, the heads of the phrases, and the correct semantic role labeling. Heads are taken to be the rightmost words of chunks. On average, there are 18.8 phrases per proposition, vs. 23.5 words per sentence. Sentences may contain multiple propositions. There are 4305 propositions in section 20.

5.1 Constraints

Roth and Yih use five global constraints on label sequences for the semantic role labeling task. We express these constraints as FSAs. The first two are general, and the seven automata encoding them can be constructed offline:

- NO DUPLICATE ARGUMENT LABELS (Fig. 4(a)) requires that each verb have at most one argument of each type in a given sentence. We separate this into six individual constraints, one for each core argument type. Thus, we have constraints called NO DUPLICATE A0, NO DUPLICATE A1, etc. Each of these is represented as a three-state FSA.

- AT LEAST ONE ARGUMENT (Fig. 1) simply requires that the label sequence is not O^*. This is a two-state automaton as described in §2.

The last three constraints require information about the example, and the automata must be constructed on a per-example basis:

- ARGUMENT CANDIDATES (Fig. 5) encodes a set of position spans each of which must receive only a single label type. These spans were proposed using a high-recall heuristic (Xue and Palmer, 2004).

- KNOWN VERB POSITION (Fig. 4(b)) simply encodes the position of the verb in question, which must be labeled O.

- DISALLOW ARGUMENTS (Fig. 4(c)) specifies argument types that are compatible with the verb in question, according to PropBank.

5.2 Experiments

We implemented our hard constraint relaxation algorithm, using the FSA toolkit (Kanthak and Ney, 2004) for finite-state operations. FSA is an open-source C++ library providing a useful set of algorithms on weighted finite-state acceptors and transducers. For each example we decoded, we chose a random order in which to apply the constraints.

Lattices are generated from what amounts to a unigram model—the voted perceptron classifier of Roth and Yih. The features used are a subset of those commonly applied to the task.

Our system produces output identical to that of Roth and Yih. Table 1 shows F-measure on the core arguments. Table 2 shows a runtime comparison. The ILP runtime was provided by the authors (personal communication). Because the systems were run under different conditions, the times are not directly comparable. However, constraint relaxation is more than sixteen times faster than ILP despite running on a slower platform.

5.2.1 Comparison to an ILP solver

Roth and Yih's linear program has two kinds of numeric constraints. Some encode the shortest path problem structure; the others encode the global constraints of §5.1. The ILP solver works by relaxing to a (real-valued) linear program, which may obtain a fractional solution that represents a path mixture instead of a path. It then uses branch-and-bound to seek the optimal rounding of this fractional solution to an integer solution (Guéret et al., 2002) that represents a single path satisfying the global constraints.

Our method avoids fractional solutions: a relaxed solution is always a true single path, which either

Mr.	Turner	said	the	test	will	be	shipped	in	45	days	to	hospitals	and	clinical	laboratories	.
B-NP	I-NP	B-VP	B-NP	I-NP	B-VP	I-VP	I-VP	B-PP	B-NP	I-NP	B-PP	B-NP	O	B-NP	I-NP	O
	Turner	said		test			shipped	in		days	to	hospitals	and		laboratories	.
	A0	O		A1			A1	A1		A1	A1	A1	A1		A1	O

Figure 3: Example sentence, with phrase tags and heads, and core argument labels. The A1 argument of "said" is a long clause.

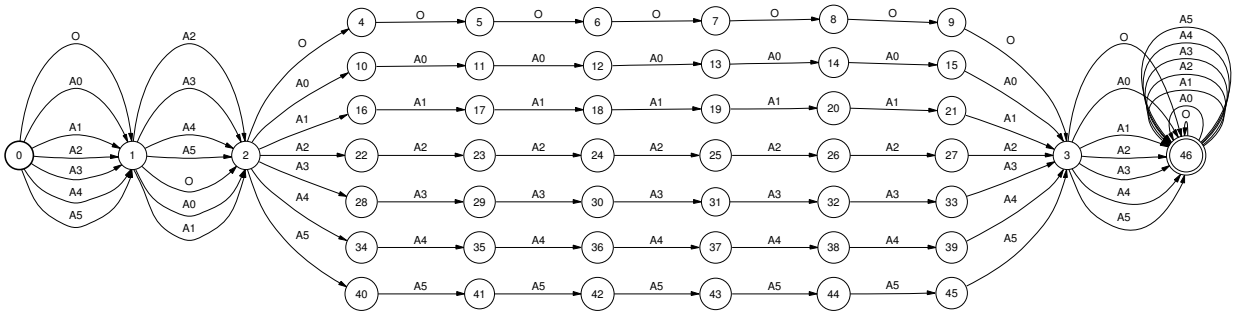


Figure 5: An automaton expressing ARGUMENT CANDIDATES.

Argument	Count	F-measure
A0	2849	79.27
A1	4029	75.59
A2	943	55.68
A3	149	46.41
A4	147	81.82
A5	4	25.00
All	8121	74.51

Table 1: F-measure on core arguments.

Constraint	Violations	Fraction
ARGUMENT CANDIDATES	1619	0.376
NO DUPLICATE A1	899	0.209
NO DUPLICATE A0	348	0.081
NO DUPLICATE A2	151	0.035
AT LEAST ONE ARGUMENT	108	0.025
DISALLOW ARGUMENTS	48	0.011
NO DUPLICATE A3	13	0.003
NO DUPLICATE A4	3	0.001
NO DUPLICATE A5	1	0.000
KNOWN VERB POSITION	0	0.000

Table 3: Violations of constraints by $\mathbf{y}_0^*$.

satisfies or violates each global constraint. In effect, we are using two kinds of domain knowledge. First, we recognize that this is a graph problem, and insist on true paths so we can use Viterbi decoding. Second, we choose to relax only domain-specific constraints that are likely to be satisfied anyway (in our domain), in contrast to the meta-constraint of integrality relaxed by ILP. Thus it is cheaper on average for us to repair a relaxed solution. (Our repair strategy—finite-state intersection in place of branch-and-bound search—remains expensive in the worst case, as the problem is NP-hard.)

5.2.2 Constraint violations

The $\mathbf{y}_0^*$s, generated with only *local* information, satisfy most of the global constraints most of the time. Table 3 shows the violations by type.

The *majority* of best labelings according to the local model don't violate *any* global constraints—a fact especially remarkable because there are *no* label sequence features in Roth and Yih's unigram model. This confirms our intuition that natural language structure is largely apparent locally. Table 4 shows the breakdown. The majority of examples are very efficient to decode, because they don't require intersection of the lattice with any constraints—$\mathbf{y}_0^*$ is extracted and is good enough. Those examples where constraints *are* violated are still relatively efficient because they only require a small number of intersections. In total, the average number of intersections needed, even with the naive randomized constraint ordering, was only 0.65. The order doesn't matter very much, since 75% of examples have one violation or fewer.

5.2.3 Effects on lattice size

Figure 6 shows the effect of intersection with violated constraints on the average size of lattices, measured in arcs. The vertical bars at $k = 0$, $k = 1, \ldots$ show the number of examples where con-

Method	Total Time	Per Example	Platform
Brute Force Finite-State	37m25.290s	0.522s	Pentium III, 1.0 GHz
ILP	11m39.220s	0.162s	Xeon, 3.x GHz
Constraint Relaxation	39.700s	0.009s	Pentium III, 1.0 GHz

Table 2: A comparison of runtimes for constrained decoding with ILP and FSA.

Violations	Labelings	Fraction	Cumulative
0	2368	0.550	0.550
1	863	0.200	0.750
2	907	0.211	0.961
3	156	0.036	0.997
4	10	0.002	0.999
5	1	0.000	1.000
6–10	0	0.000	1.000

Table 4: Number of $\mathbf{y}_0^*$ with each violation count.

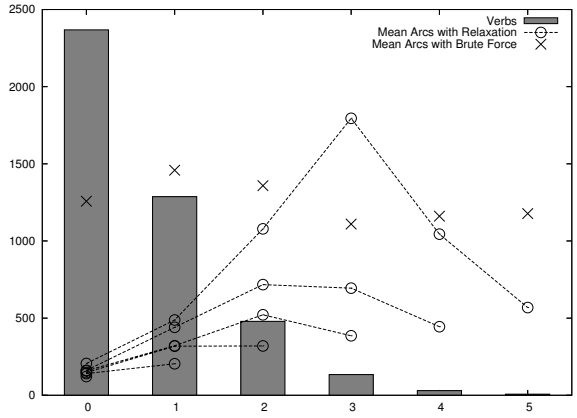


Figure 6: Mean lattice size (measured in arcs) throughout decoding. Vertical bars show the number of examples over which each mean is computed.

straint relaxation had to intersect k contraints (i.e., $\mathbf{y}^* \equiv \mathbf{y}_k^*$). The trajectory ending at (for example) $k = 3$ shows how the average lattice size for that subset of examples evolved over the 3 intersections. The X at $k = 3$ shows the final size of the brute-force lattice on the *same* subset of examples.

For the most part, our lattices do stay much smaller than those produced by the brute-force algorithm. (The uppermost curve, $k = 5$, is an obvious exception; however, that curve describes only the seven hardest examples.) Note that plotting only the final size of the brute-force lattice obscures the long trajectory of its construction, which involves 10 intersections and, like the trajectories shown, includes larger intermediate automata.[2] This explains the far

[2]The final brute-force lattice is especially shrunk by its inclusion of, for example, DISALLOW ARGUMENTS, which can only remove arcs. That constraint is rarely included in the relaxation lattices because it is rarely violated (see Table 3).

Constraint	Violations	Fraction
ARGUMENT CANDIDATES	90	0.0209
AT LEAST ONE ARGUMENT	27	0.0063
NO DUPLICATE `A2`	3	0.0007
NO DUPLICATE `A0`	2	0.0005
NO DUPLICATE `A1`	2	0.0005
NO DUPLICATE `A3`	1	0.0002
NO DUPLICATE `A4`	1	0.0002

Table 5: Violations of constraints by $\hat{\mathbf{y}}$, measured over the development set.

longer runtime of the brute-force method (Table 2).

Harder examples (corresponding to longer trajectories) have larger lattices, on average. This is partly just because it is disproportionately the longer sentences that are hard: they have more opportunities for a relaxed decoding to violate global constraints.

Hard examples are rare. The left three columns, requiring only 0–2 intersections, constitute 96% of examples. The vast majority can be decoded without much more than doubling the local-lattice size.

6 Soft constraints

The gold standard labels $\hat{\mathbf{y}}$ occasionally violate the hard global constraints that we are using. Counts for the development set appear in Table 5. Counts for violations of NO DUPLICATE `A`· do *not* include discontinous arguments, of which there are 104 instances, since we ignore them.

Because of the infrequency, the hard constraints still help most of the time. However, on a small subset of the examples, they preclude us from inferring the correct labeling.

We can apply these constraints with weights, rather than making them inviolable. This constitutes a transition from hard to soft constraints. Formally, a soft constraint C: $\mathcal{Y}^* \mapsto \mathbb{R}^-$ is a mapping from a label sequence to a non-positive penalty.

Soft constraints present new difficulty for decod-

```
SOFT-CONSTRAIN-LATTICE(L, C):
1.  (y*, Score(y*)) := (empty, −∞)
2.  branches := [(L, C, 0)]
3.  while (L, C, penalty) := Dequeue(branches):
4.      L := Prune(L, Score(y*) − penalty)
5.      unless Empty(L):
6.          y := Best-Path(L)
7.          for C ∈ C:
8.              if C(y) < 0:                    (* so C(y) = w_C *)
9.                  C := C − {C}
10.                 Enqueue(branches, (L ∩ C, C, penalty))
11.                 penalty := penalty + C(y)
12.         if Score(y*) < L(y) + penalty:
13.             (y*, Score(y*)) := (y, L(y) + penalty)
14. return y*
```

Figure 7: Soft constraints decoding algorithm

ing, because instead of eliminating paths of L from contention, they just reweight them.

In what follows, we consider only binary soft constraints—they are either satisfied or violated, and the same penalty is assessed whenever a violation occurs. That is, $\forall C \in \mathcal{C}, \exists w_C < 0$ such that $\forall \mathbf{y}, C(\mathbf{y}) \in \{0, w_C\}$.

6.1 Soft constraint relaxation

The decoding algorithm for soft constraints is a generalization of that for hard constraints. The difference is that, whereas with hard constraints a violation meant disqualification, here violation simply means a penalty. We therefore must find and compare two labelings: the best that satisfies the constraint, and the best that violates it.

We present a branch-and-bound algorithm (Lawler and Wood, 1966), with pseudocode in Figure 7. At line 9, we process and eliminate a currently violated constraint $C \in \mathcal{C}$ by considering two cases. On the first branch, we insist that C be satisfied, enqueuing $L \cap C$ for later exploration. On the second branch, we assume C is violated by all paths, and so continue considering L unmodified, but accept a penalty for doing so; we immediately explore the second branch by returning to the start of the **for** loop.[3]

Not every branch needs to be completely explored. Bounding is handled by the PRUNE function at line 4, which shrinks L by removing some or all paths that cannot score better than Score($\mathbf{y}^*$), the score of the best path found on any branch so far. Our experiments used almost the simplest possible PRUNE: replace L by the empty lattice if the best path falls below the bound, else leave L unchanged.[4]

A similar bounding would be possible in the implicit branches. If, during the **for** loop, we find that the test at line 12 would fail, we can quit the **for** loop and immediately move to the next branch in the queue at line 3.

There are two factors in this algorithm that contribute to avoiding consideration of *all* of the exponential number of leaves corresponding to the power set of constraints. First, bounding stops evaluation of subtrees. Second, only *violated* constraints require branching. If a lattice's best path satisifies a constraint, then the best path that violates it can be no better since, by assumption, $\forall \mathbf{y}, C(\mathbf{y}) \leq 0$.

6.2 Runtime experiments

Using the ten constraints from §5.1, weighted naively by their log odds of violation, the soft constraint relaxation algorithm runs in a time of 58.40 seconds. It is, as expected, slower than hard constraint relaxation, but only by a factor of about two.

As a side note, softening these particular constraints in this particular way did not improve decoding quality in this case. It might help to jointly train the relative weights of these constraints and the local model—e.g., using a perceptron algorithm (Freund and Schapire, 1998), which repeatedly extracts the best global path (using our algorithm), compares it to the gold standard, and adjusts the constraint weights. An obvious alternative is maximum-entropy training, but the partition function would have to be computed using the large brute-force lattices, or else approximated by a sampling method.

7 Future work

For a given task, we may be able to obtain further speedups by carefully choosing the order in which to test and apply the constraints. We might treat this as a reinforcement learning problem (Sutton, 1988),

[3] It is possible that a future best path on the second branch will *not* actually violate C, in which case we have overpenalized it, but in that case we will also find it with correct penalty on the first branch.

[4] Partial pruning is also possible: by running the Viterbi version of the forward-backward algorithm, one can discover for each edge the weight of the best path on which it appears. One can then remove all edges that do not appear on any sufficiently good path.

where an agent will obtain rewards by finding $\mathbf{y}^*$ quickly. In the hard-constraint algorithm, for example, the agent's possible moves are to test some constraint for violation by the current best path, or to intersect some constraint with the current lattice. Several features can help the agent choose the next move. How large is the current lattice, which constraints does it already incorporate, and which remaining constraints are already known to be satisfied or violated by its best path? And what were the answers to those questions at previous stages?

Our constraint relaxation method should be tested on problems other than semantic role labeling. For example, information extraction from bibliography entries, as discussed in §1, has about 13 fields to extract, and interesting hard and soft global constraints on co-occurrence, order, and adjacency. The method should also be evaluated on a task with longer sequences: though the finite-state operations we use do scale up linearly with the sequence length, longer sequences have more chance of violating a global constraint somewhere in the sequence, requiring us to apply that constraint explicitly.

8 Conclusion

Roth and Yih (2005) showed that global constraints can improve the output of sequence labeling models for semantic role labeling. In general, decoding under such constraints is NP-complete. We exhibited a practical approach, finite-state constraint relaxation, that greatly sped up decoding on this NLP task by using familiar finite-state operations—weighted FSA intersection and best-path extraction—rather than integer linear programming.

We have also given a constraint relaxation algorithm for binary soft constraints. This allows incorporation of constraints akin to reranking features, in addition to inviolable constraints.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0347822. We thank Scott Yih for kindly providing both the voted-perceptron classifier and runtime results for decoding with ILP, and the reviewers for helpful comments.

References

Xavier Carreras and Lluís Màrques. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proc. of CoNLL*, pp. 89–97.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proc. of ACL*, pp. 363–370.

Yoav Freund and Robert E. Schapire. 1998. Large margin classification using the perceptron algorithm. In *Proc. of COLT*, pp. 209–217, New York. ACM Press.

Christelle Guéret, Christian Prins, and Marc Sevaux. 2002. *Applications of optimization with Xpress-MP*. Dash Optimization. Translated and revised by Susanne Heipcke.

Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2004. Semantic role labeling by tagging syntactic chunks. In *Proc. of CoNLL*, pp. 110–113.

Stephan Kanthak and Hermann Ney. 2004. FSA: An efficient and flexible C++ toolkit for finite state automata using on-demand computation. In *Proc. of ACL*, pp. 510–517.

Lauri Karttunen, Jean-Pierre Chanod, Gregory Grefenstette, and Anne Schiller. 1996. Regular expressions for language engineering. *Journal of Natural Language Engineering*, 2(4):305–328.

Kimmo Koskenniemi. 1990. Finite-state parsing and disambiguation. In *Proc. of COLING*, pp. 229–232.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*, pp. 282–289.

Eugene L. Lawler and David E. Wood. 1966. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719.

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

Mehryar Mohri, Fernando Pereira, and Michael Riley. 1996. Weighted automata in text and speech processing. In A. Kornai, editor, *Proc. of the ECAI 96 Workshop*, pp. 46–50.

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

Fuchun Peng and Andrew McCallum. 2004. Accurate information extraction from research papers using conditional random fields. In *Proc. of HLT-NAACL*, pp. 329–336.

Dan Roth and Wen-tau Yih. 2005. Integer linear programming inference for conditional random fields. In *Proc. of ICML*, pp. 737–744.

Richard S. Sutton. 1988. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44.

Nathan Vaillette. 2004. *Logical Specification of Finite-State Transductions for Natural Language Processing*. Ph.D. thesis, Ohio State University.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proc. of EMNLP*, pp. 88–94.

Semantic Role Labeling of Nominalized Predicates in Chinese

Nianwen Xue
Center for Research in Spoken Language
University of Colorado
Boulder, CO, 80309
Nianwen.Xue@colorado.edu

Abstract

Recent work on semantic role labeling (SRL) has focused almost exclusively on the analysis of the predicate-argument structure of verbs, largely due to the lack of human-annotated resources for other types of predicates that can serve as training and test data for the semantic role labeling systems. However, it is well-known that verbs are not the only type of predicates that can take arguments. Most notably, nouns that are nominalized forms of verbs and relational nouns generally are also considered to have their own predicate-argument structure. In this paper we report results of SRL experiments on nominalized predicates in Chinese, using a newly completed corpus, the Chinese Nombank. We also discuss the impact of using publicly available manually annotated verb data to improve the SRL accuracy of nouns, exploiting a widely-held assumption that verbs and their nominalizations share the same predicate-argument structure. Finally, we discuss the results of applying reranking techniques to improve SRL accuracy for nominalized predicates, which showed insignificant improvement.

1 Introduction

Detecting and classifying the arguments of predicates has been an active area of research in recent years, driven by the availability of large-scale semantically annotated corpora such as the FrameNet (Baker et al., 1998) and the Propbank (Palmer et al., 2005). It is generally formulated as a semantic role labeling (SRL) task, where each argument of the predicate is assigned a label that represents the semantic role it plays with regard to its predicate (Gildea and Jurafsky, 2002; Hacioglu et al., 2003; Pradhan et al., 2004b; Xue and Palmer, 2004; Toutanova et al., 2005; Koomen et al., 2005). It has been the shared task for the CoNLL competition for two consecutive years (Carreras and Màrquez, 2004b; Carreras and Màrquez, 2005). This line of research has also expanded from English to other languages (Sun and Jurafsky, 2004; Xue and Palmer, 2005). So far, however, most of the research efforts have focused on analyzing the predicate-argument structure of verbs, largely due to absence of annotated data for other predicate types. In this paper, we report SRL experiments performed on nominalized predicates in Chinese, taking advantage of a newly completed corpus, the Chinese Nombank (Xue, 2006), which we describe in greater detail in Section 2. The rest of the paper is organized as follows. Section 3 describes the architecture of our system as well as the features we used in our experiments. In Section 4 we describe the experimental setups and report our experimental results. We first present experiments that use hand-crafted parses as input, providing a measurement of how well the Nombank annotation can be bootstrapped from the syntactic structure in the treebank. We then describe a more realistic experimental setup in which an automatic parser is first used to parse unsegmented raw

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 431–438,
New York, June 2006.

text and its output is then fed into our SRL system. We also discuss whether verb data can be used to improve the SRL accuracy of nominalized predicates. Finally we describe a preliminary experiment that uses reranking techniques to improve the SRL accuracy on hand-crafted parses. Section 5 attempts to put our results in perspective in the context of related work. Section 6 concludes our paper.

2 The Chinese Nombank

The Chinese Nombank extends the general annotation framework of the English Proposition Bank (Palmer et al., 2005) and the English Nombank (Meyers et al., 2004) to the annotation of nominalized predicates in Chinese. Like the English Nombank project, the Chinese Nombank adds a layer of semantic annotation to the Chinese TreeBank (CTB), a syntactically annotated corpus of 500 thousand words. The Chinese Nombank annotates two types of elements that are associated with the nominalized predicate: argument-like elements that are expected of this predicate, and adjunct-like elements that modify this predicate. Arguments are assigned numbered labels (prefixed by *ARG*, e.g., ARG0...ARGn) while adjuncts receive a functional tag (e.g., TMP for temporal, LOC for locative, MNR for manner) prefixed by *ARGM*. A predicate generally has no more than six numbered arguments and the complete list of functional tags for adjuncts and their descriptions can be found in the annotation guidelines of this project.

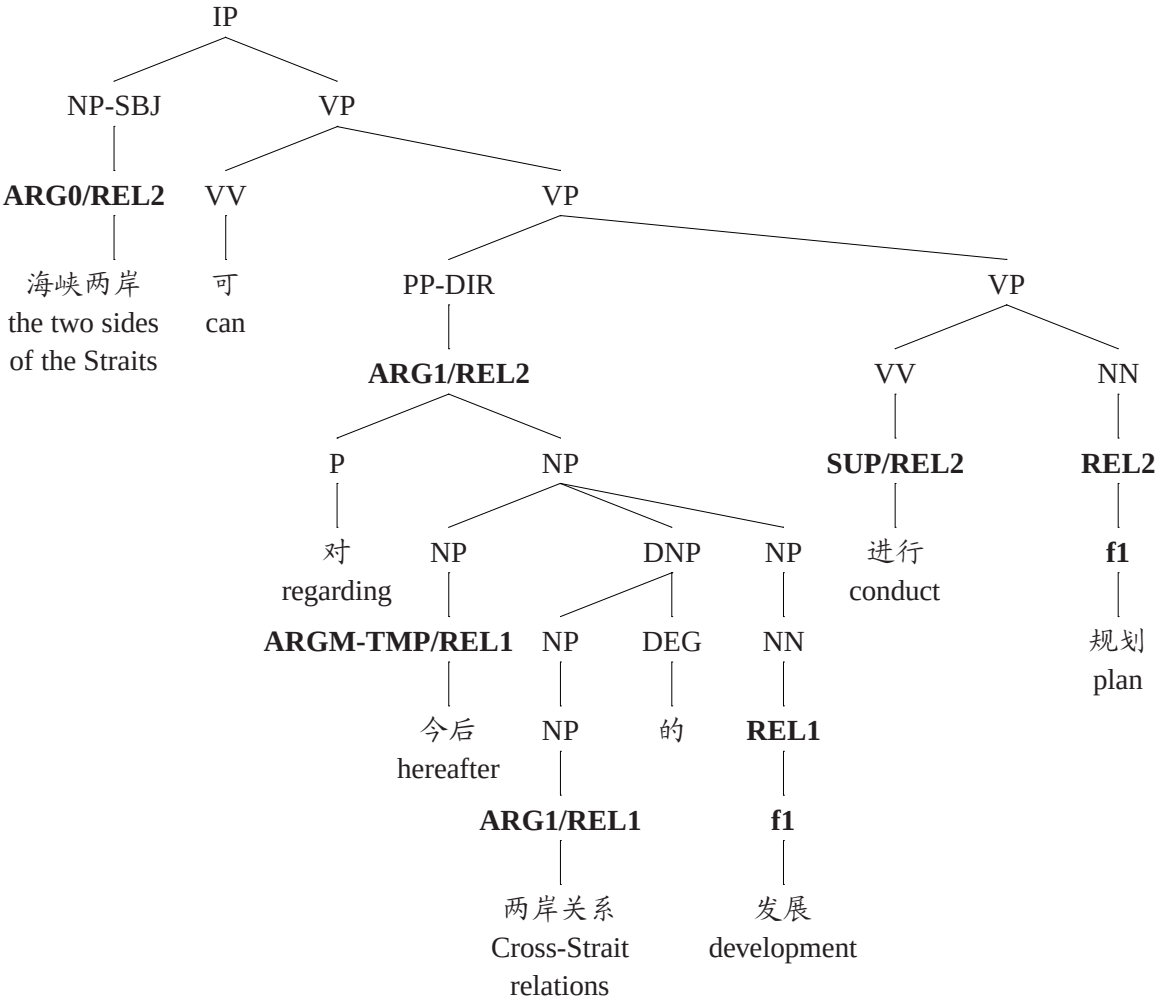
The Chinese Nombank also adds a coarse-grained sense tag to the predicate. The senses of a predicate, formally called *framesets*, are motivated by the argument structure of this predicate and are thus an integral part of the predicate-argument structure annotation. Sense disambiguation is performed only when different senses of a predicate require different sets of arguments. These senses are the same senses defined for the corresponding verbs in the Chinese Proposition Bank, but typically only a subset of the verb senses are realized in their nominalized forms. The example in 1 illustrates the Chinese Nombank annotations, which are the labels in bold in the parse tree. Take 发展(”development”) as an example, **f1** is the frameset identifier. Of the four expected arguments for this frameset, **ARG0** the cause or agent, **ARG1** the theme, **ARG2** the initial state and **ARG3** the end state or goal, only ARG1 is realized and it is 两岸关系(”cross-Strait relations”). The predicate also has a modifier labeled **ARGM-TMP**, 今后(”hereafter”).

Typically the arguments and adjuncts of a nominalized predicate are realized inside the noun phrase headed by the nominalized predicate, as is the case for 发展(“development”) in Example 1. A main exception is when the noun phrase headed by the nominalized predicate is an object of a support verb, in which case the arguments of this predicate can occur outside the noun phrase. This is illustrated by 规划(“planning”) in Example 1, where the noun phrase of which it is the head is the object of a support verb 进行(“conduct”), which has little meaning of its own. Both arguments of this predicate, 海峡两岸(“the two sides of the Taiwan Strait”) and 今后两岸关系的发展(“the development of the cross-Strait relations”), are realized outside the noun phrase. There are also a few other general tendencies about the arguments of nominalized predicates that are worth pointing out. The distribution of their arguments is much less predictable than verbs whose arguments typically occupy prominent syntactic positions like the subject and object. There also tend to be fewer arguments that are actually realized for nominalized predicates. Nominalized predicates also tend to take fewer types of adjuncts (**ARGM**s) than their verbal counterpart and they also tend to be less polysemous, having only a subset of the senses of their verb counterpart.

The goal of the semantic role labeling task described in this paper is to identify the arguments and adjuncts of nominalized predicates and assign appropriate semantic role labels to them. For the purposes of our experiments, the sense information of the predicates are ignored and left for future research.

3 System description

The predominant approach to the semantic role labeling task is to formulate it as a classification problem that can be solved with machine-learning techniques. Argument detection is generally formulated as a binary classification task that separates constituents that are arguments or adjuncts to a pred-



The two sides of the Taiwan Straits can plan the development of the cross-Strait relations hereafter.

Table 1: A nominalized predicate annotated with semantic roles

icate from those that are not related to the predicate in question. Argument classification, which classifies the constituents into a category that corresponds to one of the argument or adjunct labels is a natural multi-category classification problem. Many classification techniques, SVM (Pradhan et al., 2004b), perceptrons (Carreras and Màrquez, 2004a), Maximum Entropy (Xue and Palmer, 2004), etc. have been successfully used to solve SRL problems. For our purposes here, we use a Maximum Entropy classifier with a tunable Gaussian prior in the Mallet Toolkit[1]. The Maximum Entropy classifier does multi-category classification and thus can be straightforwardly applied to the problem here. The classifier can be tuned to minimize overfitting by adjusting the Gaussian prior.

3.1 A three-stage architecture

Like verbal predicates, the arguments and adjuncts of a nominalized predicate are related to the predicate itself in linguistically well-understood structural configurations. As we pointed out in Section 2, most of the arguments for nominalized predicates are inside the NP headed by the predicate unless the NP is the object of a support verb, in which case its arguments can occur outside the NP. Typically the subject of the support verb is also an argument of the nominalized predicate, as illustrated in Example 1.

[1] http://mallet.cs.umass.edu

The majority of the constituents are not related to the predicate in question, especially since the sentences in the treebank tend to be very long. This is clearly a lingustic observation that can be exploited for the purpose of argument detection. There are two common approaches to argument detection in the SRL literature. One is to apply a binary classifier directly to all the constituents in the parse tree to separate the arguments from non-arguments, and let the machine learning algorithm do the work. This can be done with high accuracy when the machine-learning algorithm is powerful and is provided with appropriate features (Hacioglu et al., 2003; Pradhan et al., 2004b). The alternative approach is to combine heuristic and machine-learning approaches (Xue and Palmer, 2004). Some negative samples are first filtered out with heuristics that exploit the syntactic structures represented in a parse tree before a binary classifier is applied to further separate the positive samples from the negative samples. It turns out the heuristics that are first proposed in Xue and Palmer (2004) to prune out non-arguments for verbal predicates can be easily adapted to detect arguments for the nominalized predicates as well, so in our experiments we adopt the latter approach. The algorithm starts from the predicate that anchors the annotation, and first collects all the sisters of this predicate. It then iteratively moves one level up to the parent of the current node to collect its sisters till it reaches the appropriate top-level node. At each level, the system has a procedure to determine whether that level is a coordination structure or a modification structure. The system only considers a constituent to be a potential candidate if it is an adjunct to the current node. Punctuation marks at all levels are skipped. After this initial procedure, a binary classifier is applied to distinguish the positive samples from the negative samples. A lower threshold is used for positive samples than negative samples to maximize the recall so that we can pass along as many positive samples as possible to the next stage, which is the multi-category classification.

3.2 Features

SRL differs from low-level NLP tasks such as POS tagging in that it has a fairly large feature space and as a result linguistic knowledge is crucial in designing effective features for this task. A wide range of features have been shown to be useful in previous work on semantic role labeling for verbal predicates (Gildea and Jurafsky, 2002; Pradhan et al., 2004b; Xue and Palmer, 2004) and our experiments show most of them are also effective for SRL of nominalized predicates. The features for our multicategory classifier are listed below:

- *Predicate*: The nominalized predicate itself.
- *Position*: The position is defined in relation to the predicate and the values are *before* and *after*. Since most of the arguments for nominalized predicates in Chinese are before the predicates, this feature is not as effective as when it is used for verbal predicates.
- *path*: The path between the constituent being classified and the predicate.
- *path + dominating verb*. The path feature combined with the dominating verb. This feature is only invoked when there is an intervening dominating verb between the constituent being classified and the predicate. It is used to capture the observation that only a closed set of verbs can be support verbs for nominalized predicates and they are good indicators of whether or not the constituent is an argument of this predicate and the semantic role of the argument.
- *Head word and its part of speech*: The head word and its part-of-speech have proved to be a good indicator of the semantic role label of a constituent for verbal predicates in previous work. It proves to be a good feature for nominal predicates as well.
- *Phrase type*: The syntactic category of the constituent being classified.
- *First and last word of the constituent being classified*
- *sisterhood with predicate*: A binary feature that indicates whether the constituent being classified is a sister to the nominalized predicate.
- *Combination features*: predicate-head word combination, predicate-phrase type combination.

- *class features*. Features that replace the predicate with its class. The class features are induced from frame files through a procedure first introduced in (Xue and Palmer, 2005).

Not all the features used for multicategory classification are equally effective for binary classification, which only determines whether or not a constituent is an argument or adjunct to the nominalized predicate. Therefore, the features for the binary classifier are a subset of the features used for multicategory classification. These are path, path plus dominating verb, head word and its part-of-speech and sisterhood.

4 Experiments

4.1 Data

Our system is trained and tested on a pre-release version of the Chinese Nombank. This version of the Chinese Nombank consists of standoff annotation on the first 760 articles (`chtb_001.fid` to `chtb_931.fid`) of the Penn Chinese Treebank[2]. This chunk of data has 250K words and 10,364 sentences. It has 1,227 nominalized predicate types and 10,497 nominalized predicate instances. In comparison, there are 4,854 verb predicate types and 37,183 verb predicate instances in the same chunk of data. By instance, the size of the Nombank is between a quarter and one third of the Chinese Proposition Bank. Following the convention of the semantic role labeling experiments in previous work, we divide the training and test data by the number of articles, not by the predicate instances. This pretty much guarantees that there will be unseen predicates in the test data. For all our experiments, 688 files are used as training data and the other 72 files (`chtb_001.fid` to `chtb_040.fid` and `chtb_900.fid` to `chtb_931.fid`) are held out as test data. The test data is selected from the double-annotated files in the Chinese Treebank and the complete list of double-annotated files can be found in the documentation for the Chinese Treebank 5.1. Our parser is trained and tested with the same data partition as our semantic role labeling system.

[2]The most current version (CTB5.1) of the Penn Chinese Treebank has 507K words, 825K Chinese characters, 18,716 sentences and 890 articles.

4.2 Semantic role tagging with hand-crafted parses

In this section we present experimental results using Gold Standard parses in the Chinese Treebank as input. To be used in real-world natural language applications, a semantic role tagger has to use automatically produced constituent boundaries either from a parser or by some other means, but experiments with Gold Standard input will help us evaluate how much of a challenge it is to map a syntactic representation to a semantic representation, which may very well vary from language to language. There are two experimental setups. In the first experiment, we assume that the constituents that are arguments or adjuncts are known. We only need to assign the correct argument or adjunct labels. In the second experiment, we assume that all the constituents in a parse tree are possible arguments. The system first filters out consituents that are highly unlikely to be an argument for the predicate, using the heuristics described in Section 3. A binary classifier is then applied to the remaining constituents to do further separation. Finally the multicategory classifier is applied to the candidates that the binary classifier passes along. The results of these two experiments are presented in Table 2.

experiments	all			core
	p (%)	r(%)	f(%)	f(%)
constituents known	n/a	n/a	86.6	86.9
constituents unknown	69.7	73.7	71.6	72.0

Table 2: Results for hand-crafted parses

Compared with the 93.9% reported by Xue and Palmer (2005) for verbal predicates on the same data, the 86.9% the system achieved when the consituents are given is considerably lower, suggesting that SRL for nominalized predicates is a much more challenging task. The difference between the SRL accuracy for verbal and nominalized predicates is even greater when the constituents are not given and the system has to identify the arguments to be classified. Xue and Palmer reported an f-score of 91.4% for verbal predicates under similar experimental conditions, in contrast with the 71.6% our system achieved for nominalized predicates. Careful error analysis shows that one important cause for

this degradation in performance is the fact that there is insufficient training data for the system to reliably separate support verbs from other verbs and determine whether the constituents outside the NP headed by the nominalized predicate are related to the predicate or not.

4.3 Using automatic parses

We also conducted an experiment that assumes a more realistic scenario in which the input is raw unsegmented text. We use a fully automatic parser that integrates segmentation, POS tagging and parsing. Our parser is similar to (Luo, 2003) and is trained and tested on the same data partition as the semantic role labeling system. Tested on the held-out test data, the labeled precision and recall are 83.06% and 80.15% respectively for all sentences. The results are comparable with those reported in Luo (Luo, 2003), but they cannot be directly compared with most of the results reported in the literature, where correct segmentation is assumed. In addition, in order to account for the differences in segmentation, each character has to be treated as a leaf of the parse tree. This is in contrast with word-based parsers where words are terminals. Since semantic role tagging is performed on the output of the parser, only constituents in the parse tree are candidates. If there is no constituent in the parse tree that shares the same text span with an argument in the manual annotation, the system cannot possibly get a correct annotation. In other words, the best the system can do is to correctly label all arguments that have a constituent with the same text span in the parse tree.

all			core
p (%)	r(%)	f(%)	f(%)
49.7	53.1	51.3	48.3

Table 3: Results for automatic parses

The results show a similar performance degradation compared with the results reported for verbs on the same data in previous work, which is not unexpected. Xue and Palmer (2005) reported an f-score of 61.3% when a parser is used to preprocess the data.

4.4 Using verb data to improve noun SRL accuracy

Since verbs and their nominalized counterparts are generally considered to share the same argument structure and in fact the Chinese Nombank is annotated based on the same set of lexical guidelines (called *frame files*) as the Chinese PropBank, it seems reasonable to expect that adding the verb data to the training set will improve the SRL accuracy of the nominal predicates, especially when the training set is relatively small. Given that verbs and their nominalized counterpart share the same morphological form in Chinese, adding the verb data to the training set is particularly straightforward. In our experiments, we extracted verb instances from the CPB that have nominalized forms in the portion of the Chinese Treebank on which our SRL experiments are performed and added them to the training set. Our experiments show, however, that simply adding the verb data to the training set and indiscriminately extracting the same features from the verb and noun instances will hurt the overall performance instead of improving it. This result is hardly surprising upon closer examination: the values of certain features are vastly different for verbal and nominal predicates. Most notably, the path from the predicate to the constituent being classified, an important feature for semantic role labeling systems, differ greatly from nominal and verbal predicates. When they are thrown in the same training data mix, they effectively create noise and neutralize the discriminative effect of this feature. Other features, such as the head words and their POS tags, are the same and adding these features does indeed improve the SRL accuracy of nominal predicates, although the improvement is not statistically significant.

4.5 Reranking

In a recent paper on the SRL on verbal predicates for English, (Toutanova et al., 2005) pointed out that one potential flaw in a SRL system where each argument is considered on its own is that it does not take advantage of the fact that the arguments (not the adjuncts) of a predicate are subject to the hard constraint that they do not have the same label[3]. They

[3]For certain symmetrical predicates, arguments can have the same label, although these cases are rare.

show that by performing joint learning of all the arguments in the same proposition (for the same predicate), the SRL accuracy is improved. To test the efficacy of joint-learning for nominalized predicates in Chinese, we conducted a similar experiment, using a perceptron reranker described in Shen and Joshi (2004). Arguments and adjuncts of the same predicate instance (proposition) are chained together with their joint probability being the product of the individual arguments and the top K propositions are selected as the reranking candidates. When the arguments are given and the input is hand-crafted gold-standard parses in the treebank, selecting the top 10 propositions yields an oracle score of 97%. This initial promise does not pan out, however. Performing reranking on the top 10 propositions did not lead to significant improvement, using the five feature classes described in (Haghighi et al., 2005). These are features that are hard to implement for individual arguments: *core argument label sequence, flattened core argument label sequence, core argument labels and phrase type sequence, repeated core argument labels with phrase types, repeated core argument labels with phrase types and adjacency information*. We speculate that the lack of improvement is due to the fact that the constraint that core (numbered) arguments should not have the same semantic role label for Chinese nominalized predicates is not as rigid as it is for English verbs. However further error analysis is needed to substantiate this speculation.

5 Related Work

Compared with large body of work on the SRL of verbal predicates, there has been relatively little work done in analyzing the predicate-argument structure of nominalized predicates. There are even less work done for the nominalized predicates for Chinese. (Hull and Comez, 1996) implemented a rule-based system for identifying the arguments for nominal predicates and (Lapata, 2002) has a system that interprets the relation between the head of noun compound and its head, but no meaningful comparison can be made between our work and theirs. Perhaps the closest work to that of ours is that of (Pradhan et al., 2004a), where they reported preliminary work for analyzing the predicate-argument structure of Chinese nominalizations, using a small data set of 630 proposition for 22 nominalizations taken from the Chinese Treebank. Since different data sets are used, the results cannot be meaningfully compared.

The results reported here for nominalized predicates are consistent with what Xue and Palmer (2005) reported for the SRL of Chinese verbs with regard to the role of the parser in their semantic role labeling system: there is a substantial performance drop when the automatic parser is used. At present, improvement in Chinese parsing is hindered by insufficient training material. Although the Chinese Treebank has a decent size of 500K words, it is evenly divided into two portions of very different sources, Xinhua newswire from mainland China and Sinorama magazines from Taiwan. Due to their very different styles, training on one portion of the data does not help or even hurt the parsing accuracy of the other portion. The lack of sufficient training material is compounded by inherent properties of the Chinese language that makes Chinese parsing particularly difficult. Chinese segmentation is a much more difficult problem than tokenization of English text and Chinese words do not have morphological clues that can help parsing decisions. We believe further improvement in SRL accuracy will be to a large extent contingent on the parsing accuracy, which requires more training material.

6 Conclusion and future work

We reported first results on the semantic role labeling of nominalized predicates in Chinese, using a sizable annotated corpus, the Chinese Nombank, as training and test data. Compared with that of verbal predicates, SRL of nominalized predicates generally presents a more challenging problem, for all experimental conditions. While the smaller training set compared with that of verbal predicates may provide partial explanation for the degradation in performance, we believe another important reason is that the arguments for nominalized predicates do not occupy prominent syntactic positions such as the subject and object, as arguments of verbal predicates often do. As a result, the syntactic structure represented in the parse tree does not provide as much of a clue for their detection and classification. However, this makes SRL of nominalized predicates a more pressing issue to solve, as they represent a substan-

tial proportion of the predicates in the corpus. Our results also show that the k-best propositions produced by the local classifier have a very high oracle score, which perhaps indicates a promising path that deserves further exploration, based on careful analysis of the errors. We intend to continue to experiment with new features and parameters for the reranking algorithm.

7 Acknowledgement

I would like to thank Martha Palmer for her unwavering support for this line of research. This work is funded by the NSF ITR via grant 130-1303-4-541984-XXXX-2000-1070.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of COLING/ACL*, pages 86–90, Montreal, Canada.

Xavier Carreras and Lluís Màrquez. 2004a. Hierarchical Recognition of Propositional Arguments with Perceptrons. In *Proceedings of the Eighth Conference on Natural Language Learning*, Boston, Massachusetts.

Xavier Carreras and Lluís Màrquez. 2004b. Introduction to the CoNLL-2004 Shared Task: Semantic Role Labeling. In *Proceedings of the Eighth Conference on Natural Language Learning*, Boston, Massachusetts.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of the Nineth Conference on Natural Language Learning*, Ann Arbor, Michigan.

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

Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2003. Shallow Semantic Parsing Using Support Vector Machines. Technical Report CSLR-2003-1, Center for Spoken Language Research at the University of Colorado.

Aria Haghighi, Kristina Toutanova, and Christopher Manning. 2005. A Joint Model for Semantic Role Labeling. In *Proceedings of the Nineth Conference on Natural Language Learning*, Ann Arbor, Michigan.

Richard D. Hull and Fernando Comez. 1996. Semantic interpretation of nominalizations. In *The AAAI Conference*, pages 1062–1068, Oregon.

Peter Koomen, Vasin Punyakanok, Dan Roth, and Wen tau Yih. 2005. Generalized Inference with Multiple Semantic Role Labeling Systems. In *Proceedings of the Nineth Conference on Natural Language Learning*, Ann Arbor, Michigan.

Maria Lapata. 2002. The disambiguation of nominalizations. *Computational Linguistics*, 28(3):357–388.

Xiaoqiang Luo. 2003. A Maximum Entropy Chinese Character-Based Parser. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP 2003)*, Sapporo, Japan.

A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. The NomBank Project: An Interim Report. In *Proceedings of the NAACL/HLT Workshop on Frontiers in Corpus Annotation*, Boston, Massachusetts.

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

Sameer Pradhan, Honglin Sun, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2004a. Parsing Arguments of Nominalizations in English and Chinese. In *Proceedings of NAACL-HLT 2004*, Boston, Mass.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James H. Martin, and Daniel Jurafsky. 2004b. Shallow Semantic Parsing Using Support Vector Machines. In *Proceedings of NAACL-HLT 2004*, Boston, Mass.

Libin Shen and Aravind K. Joshi. 2004. Flexible Margin Selection for Reranking with Full Pairwise Samples. In *Proceedings of IJCNLP-2004*, pages 446–455.

Honglin Sun and Daniel Jurafsky. 2004. Shallow Semantic Parsing of Chinese. In *Proceedings of NAACL 2004*, Boston, USA.

Kristina Toutanova, Aria Haghighi, and Christopher Manning. 2005. Joint Learning Improves Semantic Role Labeling. In *Proceedings of ACL-2005*.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for Semantic Role Labeling. In *Proceedings of 2004 Conference on Empirical Methods in Natural Language Processing*, Barcelona, Spain.

Nianwen Xue and Martha Palmer. 2005. Automatic Semantic Role Labeling for Chinese verbs. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland.

Nianwen Xue. 2006. Annotating the predicate-argument structure of Chinese nominalizations. In *Proceedings of the fifth international conference on Language Resources and Evaluation*, Genoa, Italy.

Learning for Semantic Parsing with Statistical Machine Translation

Yuk Wah Wong and Raymond J. Mooney
Department of Computer Sciences
The University of Texas at Austin
1 University Station C0500
Austin, TX 78712-0233, USA
{ywwong,mooney}@cs.utexas.edu

Abstract

We present a novel statistical approach to semantic parsing, WASP, for constructing a complete, formal meaning representation of a sentence. A semantic parser is learned given a set of sentences annotated with their correct meaning representations. The main innovation of WASP is its use of state-of-the-art statistical machine translation techniques. A word alignment model is used for lexical acquisition, and the parsing model itself can be seen as a syntax-based translation model. We show that WASP performs favorably in terms of both accuracy and coverage compared to existing learning methods requiring similar amount of supervision, and shows better robustness to variations in task complexity and word order.

1 Introduction

Recent work on natural language understanding has mainly focused on shallow semantic analysis, such as semantic role labeling and word-sense disambiguation. This paper considers a more ambitious task of *semantic parsing*, which is the construction of a complete, formal, symbolic, *meaning representation* (MR) of a sentence. Semantic parsing has found its way in practical applications such as natural-language (NL) interfaces to databases (Androutsopoulos et al., 1995) and advice taking (Kuhlmann et al., 2004). Figure 1 shows a sample MR written in a *meaning-representation language* (MRL) called CLANG, which is used for

```
((bowner our {4})
 (do our {6} (pos (left (half our)))))
```

If our player 4 has the ball, then our player 6 should stay in the left side of our half.

Figure 1: A meaning representation in CLANG

encoding coach advice given to simulated soccer-playing agents (Kuhlmann et al., 2004).

Prior research in semantic parsing has mainly focused on relatively simple domains such as ATIS (Air Travel Information Service) (Miller et al., 1996; Papineni et al., 1997; Macherey et al., 2001), in which a typcial MR is only a single semantic frame. Learning methods have been devised that can generate MRs with a complex, nested structure (cf. Figure 1). However, these methods are mostly based on deterministic parsing (Zelle and Mooney, 1996; Kate et al., 2005), which lack the robustness that characterizes recent advances in statistical NLP. Other learning methods involve the use of fully-annotated augmented parse trees (Ge and Mooney, 2005) or prior knowledge of the NL syntax (Zettlemoyer and Collins, 2005) in training, and hence require extensive human efforts when porting to a new domain or language.

In this paper, we present a novel statistical approach to semantic parsing which can handle MRs with a nested structure, based on previous work on semantic parsing using transformation rules (Kate et al., 2005). The algorithm learns a semantic parser given a set of NL sentences annotated with their correct MRs. It requires no prior knowledge of the NL syntax, although it assumes that an unambiguous, context-free grammar (CFG) of the target MRL is available. The main innovation of this al-

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 439–446,
New York, June 2006.

```
answer(count(city(loc_2(countryid(usa)))))
```
How many cities are there in the US?

Figure 2: A meaning representation in GEOQUERY

gorithm is its integration with state-of-the-art statistical machine translation techniques. More specifically, a statistical word alignment model (Brown et al., 1993) is used to acquire a bilingual lexicon consisting of NL substrings coupled with their translations in the target MRL. Complete MRs are then formed by combining these NL substrings and their translations under a parsing framework called the synchronous CFG (Aho and Ullman, 1972), which forms the basis of most existing statistical syntax-based translation models (Yamada and Knight, 2001; Chiang, 2005). Our algorithm is called WASP, short for *Word Alignment-based Semantic Parsing*. In initial evaluation on several real-world data sets, we show that WASP performs favorably in terms of both accuracy and coverage compared to existing learning methods requiring the same amount of supervision, and shows better robustness to variations in task complexity and word order.

Section 2 provides a brief overview of the domains being considered. In Section 3, we present the semantic parsing model of WASP. Section 4 outlines the algorithm for acquiring a bilingual lexicon through the use of word alignments. Section 5 describes a probabilistic model for semantic parsing. Finally, we report on experiments that show the robustness of WASP in Section 6, followed by the conclusion in Section 7.

2 Application Domains

In this paper, we consider two domains. The first domain is ROBOCUP. ROBOCUP (`www.robocup.org`) is an AI research initiative using robotic soccer as its primary domain. In the ROBOCUP Coach Competition, teams of agents compete on a simulated soccer field and receive coach advice written in a formal language called CLANG (Chen et al., 2003). Figure 1 shows a sample MR in CLANG.

The second domain is GEOQUERY, where a functional, variable-free query language is used for querying a small database on U.S. geography (Zelle and Mooney, 1996; Kate et al., 2005). Figure 2 shows a sample query in this language. Note that both domains involve the use of MRs with a complex, nested structure.

3 The Semantic Parsing Model

To describe the semantic parsing model of WASP, it is best to start with an example. Consider the task of translating the sentence in Figure 1 into its MR in CLANG. To achieve this task, we may first analyze the syntactic structure of the sentence using a *semantic grammar* (Allen, 1995), whose non-terminals are the ones in the CLANG grammar. The meaning of the sentence is then obtained by combining the meanings of its sub-parts according to the semantic parse. Figure 3(a) shows a possible partial semantic parse of the sample sentence based on CLANG non-terminals (UNUM stands for uniform number). Figure 3(b) shows the corresponding CLANG parse from which the MR is constructed.

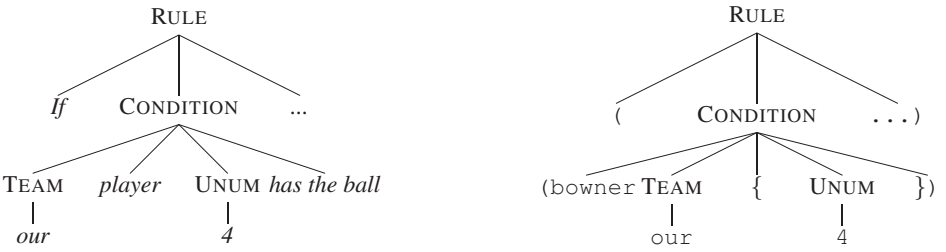
This process can be formalized as an instance of *synchronous parsing* (Aho and Ullman, 1972), originally developed as a theory of compilers in which syntax analysis and code generation are combined into a single phase. Synchronous parsing has seen a surge of interest recently in the machine translation community as a way of formalizing syntax-based translation models (Melamed, 2004; Chiang, 2005). According to this theory, a semantic parser defines a *translation*, a set of pairs of strings in which each pair is an NL sentence coupled with its MR. To finitely specify a potentially infinite translation, we use a *synchronous context-free grammar* (SCFG) for generating the pairs in a translation. Analogous to an ordinary CFG, each SCFG rule consists of a single non-terminal on the left-hand side (LHS). The right-hand side (RHS) of an SCFG rule is a pair of strings, $\langle \alpha, \beta \rangle$, where the non-terminals in β are a permutation of the non-terminals in α. Below are some SCFG rules that can be used for generating the parse trees in Figure 3:

RULE → ⟨ *if* CONDITION[1] , DIRECTIVE[2] . , (CONDITION[1] DIRECTIVE[2]) ⟩

CONDITION → ⟨ TEAM[1] *player* UNUM[2] *has the ball* , (`bowner` TEAM[1] {UNUM[2]}) ⟩

TEAM → ⟨ *our* , `our` ⟩

UNUM → ⟨ *4* , `4` ⟩



(a) English (b) CLANG

Figure 3: Partial parse trees for the CLANG statement and its English gloss shown in Figure 1

Each SCFG rule $X \rightarrow \langle \alpha, \beta \rangle$ is a combination of a production of the NL semantic grammar, $X \rightarrow \alpha$, and a production of the MRL grammar, $X \rightarrow \beta$. Each rule corresponds to a *transformation rule* in Kate et al. (2005). Following their terminology, we call the string α a *pattern*, and the string β a *template*. Non-terminals are indexed to show their association between a pattern and a template. All derivations start with a pair of associated start symbols, $\langle S_{\boxed{1}}, S_{\boxed{1}} \rangle$. Each step of a derivation involves the rewriting of a pair of associated non-terminals in both of the NL and MRL streams. Below is a derivation that would generate the sample sentence and its MR simultaneously: (Note that RULE is the start symbol for CLANG)

$\langle$RULE$_{\boxed{1}}$, RULE$_{\boxed{1}}\rangle$
$\Rightarrow \langle$*if* CONDITION$_{\boxed{1}}$, DIRECTIVE$_{\boxed{2}}$. ,
(CONDITION$_{\boxed{1}}$ DIRECTIVE$_{\boxed{2}}$)$\rangle$
$\Rightarrow \langle$*if* TEAM$_{\boxed{1}}$ *player* UNUM$_{\boxed{2}}$ *has the ball,* DIR$_{\boxed{3}}$. ,
((bowner TEAM$_{\boxed{1}}$ {UNUM$_{\boxed{2}}$}) DIR$_{\boxed{3}}$)$\rangle$
$\Rightarrow \langle$*if our player* UNUM$_{\boxed{1}}$ *has the ball,* DIR$_{\boxed{2}}$. ,
((bowner our {UNUM$_{\boxed{1}}$}) DIR$_{\boxed{2}}$)$\rangle$
$\Rightarrow \langle$*if our player 4 has the ball,* DIRECTIVE$_{\boxed{1}}$. ,
((bowner our {4}) DIRECTIVE$_{\boxed{1}}$)$\rangle$
$\Rightarrow$ ...
$\Rightarrow \langle$*if our player 4 has the ball, then our player 6 should stay in the left side of our half.* ,
((bowner our {4})
(do our {6} (pos (left (half our)))))$\rangle$

Here the MR string is said to be a *translation of* the NL string. Given an input sentence, $\mathbf{e}$, the task of semantic parsing is to find a derivation that yields $\langle \mathbf{e}, \mathbf{f} \rangle$, so that $\mathbf{f}$ is a translation of $\mathbf{e}$. Since there may be multiple derivations that yield $\mathbf{e}$ (and thus multiple possible translations of $\mathbf{e}$), a mechanism must be devised for discriminating the correct derivation from the incorrect ones.

The semantic parsing model of WASP thus consists of an SCFG, G, and a probabilistic model, parameterized by λ, that takes a possible derivation, $\mathbf{d}$, and returns its likelihood of being correct given an input sentence, $\mathbf{e}$. The output translation, $\mathbf{f}^\star$, for a sentence, $\mathbf{e}$, is defined as:

$$\mathbf{f}^\star = m\left(\arg\max_{\mathbf{d} \in D(G|\mathbf{e})} \Pr_\lambda(\mathbf{d}|\mathbf{e})\right) \quad (1)$$

where $m(\mathbf{d})$ is the MR string that a derivation $\mathbf{d}$ yields, and $D(G|\mathbf{e})$ is the set of all possible derivations of G that yield $\mathbf{e}$. In other words, the output MR is the yield of the most probable derivation that yields $\mathbf{e}$ in the NL stream.

The learning task is to induce a set of SCFG rules, which we call a *lexicon*, and a probabilistic model for derivations. A lexicon defines the set of derivations that are possible, so the induction of a probabilistic model first requires a lexicon. Therefore, the learning task can be separated into two sub-tasks: (1) the induction of a lexicon, followed by (2) the induction of a probabilistic model. Both sub-tasks require a training set, $\{\langle \mathbf{e}_i, \mathbf{f}_i \rangle\}$, where each training example $\langle \mathbf{e}_i, \mathbf{f}_i \rangle$ is an NL sentence, $\mathbf{e}_i$, paired with its correct MR, $\mathbf{f}_i$. Lexical induction also requires an unambiguous CFG of the MRL. Since there is no lexicon to begin with, it is not possible to include correct derivations in the training data. This is unlike most recent work on syntactic parsing based on gold-standard treebanks. Therefore, the induction of a probabilistic model for derivations is done in an unsupervised manner.

4 Lexical Acquisition

In this section, we focus on lexical learning, which is done by finding optimal *word alignments* between

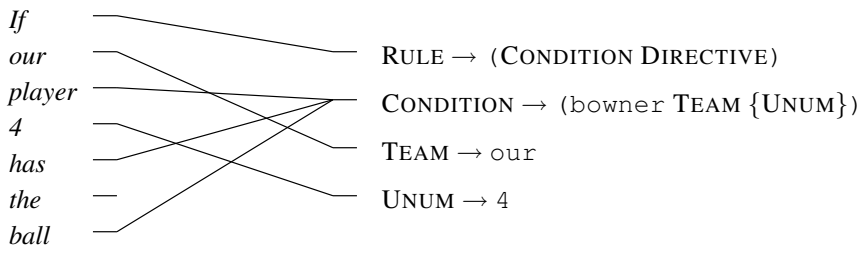


Figure 4: Partial word alignment for the CLANG statement and its English gloss shown in Figure 1

NL sentences and their MRs in the training set. By defining a mapping of words from one language to another, word alignments define a bilingual lexicon. Using word alignments to induce a lexicon is not a new idea (Och and Ney, 2003). Indeed, attempts have been made to directly apply machine translation systems to the problem of semantic parsing (Papineni et al., 1997; Macherey et al., 2001). However, these systems make no use of the MRL grammar, thus allocating probability mass to MR translations that are not even syntactically well-formed. Here we present a lexical induction algorithm that guarantees syntactic well-formedness of MR translations by using the MRL grammar.

The basic idea is to train a statistical word alignment model on the training set, and then form a lexicon by extracting transformation rules from the $K = 10$ most probable word alignments between the training sentences and their MRs. While NL words could be directly aligned with MR tokens, this is a bad approach for two reasons. First, not all MR tokens carry specific meanings. For example, in CLANG, parentheses and braces are delimiters that are semantically vacuous. Such tokens are not supposed to be aligned with any words, and inclusion of these tokens in the training data is likely to confuse the word alignment model. Second, MR tokens may exhibit polysemy. For instance, the CLANG predicate `pt` has three meanings based on the types of arguments it is given: it specifies the xy-coordinates (e.g. `(pt 0 0)`), the current position of the ball (i.e. `(pt ball)`), or the current position of a player (e.g. `(pt our 4)`). Judging from the `pt` token alone, the word alignment model would not be able to identify its exact meaning.

A simple, principled way to avoid these difficulties is to represent an MR using a sequence of productions used to generate it. Specifically, the sequence corresponds to the top-down, left-most derivation of an MR. Figure 4 shows a partial word alignment between the sample sentence and the linearized parse of its MR. Here the second production, CONDITION → `(bowner` TEAM {UNUM}`)`, is the one that rewrites the CONDITION non-terminal in the first production, RULE → `(`CONDITION DIRECTIVE`)`, and so on. Note that the structure of a parse tree is preserved through linearization, and for each MR there is a unique linearized parse, since the MRL grammar is unambiguous. Such alignments can be obtained through the use of any off-the-shelf word alignment model. In this work, we use the GIZA++ implementation (Och and Ney, 2003) of IBM Model 5 (Brown et al., 1993).

Assuming that each NL word is linked to at most one MRL production, transformation rules are extracted in a bottom-up manner. The process starts with productions whose RHS is all terminals, e.g. TEAM → `our` and UNUM → `4`. For each of these productions, $X \rightarrow \beta$, a rule $X \rightarrow \langle \alpha, \beta \rangle$ is extracted such that α consists of the words to which the production is linked, e.g. TEAM → ⟨*our*, `our`⟩, UNUM → ⟨*4*, `4`⟩. Then we consider productions whose RHS contains non-terminals, i.e. predicates with arguments. In this case, an extracted pattern consists of the words to which the production is linked, as well as non-terminals showing where the arguments are realized. For example, for the `bowner` predicate, the extracted rule would be CONDITION → ⟨TEAM$_{\boxed{1}}$ *player* UNUM$_{\boxed{2}}$ *has* (1) *ball*, `(bowner` TEAM$_{\boxed{1}}$ {UNUM$_{\boxed{2}}$}`)`⟩, where (1) denotes a *word gap* of size 1, due to the unaligned word *the* that comes between *has* and *ball*. A word gap, (g), can be seen as a non-terminal that expands to at most g words in the NL stream, which allows for some flexibility in pattern matching. Rule extraction thus proceeds backward from the end of a linearized MR

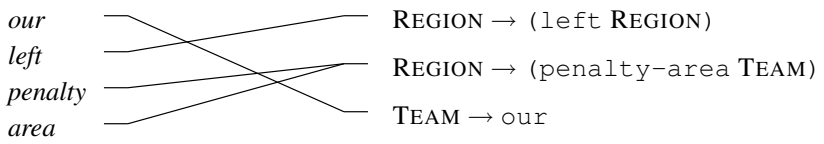


Figure 5: A word alignment from which no rules can be extracted for the penalty-area predicate

parse (so that a predicate is processed only after its arguments have all been processed), until rules are extracted for all productions.

There are two cases where the above algorithm would not extract any rules for a production r. First is when no descendants of r in the MR parse are linked to any words. Second is when there is a link from a word w, covered by the pattern for r, to a production r' outside the sub-parse rooted at r. Rule extraction is forbidden in this case because it would destroy the link between w and r'. The first case arises when a component of an MR is not realized, e.g. assumed in context. The second case arises when a predicate and its arguments are not realized close enough. Figure 5 shows an example of this, where no rules can be extracted for the penalty-area predicate. Both cases can be solved by merging nodes in the MR parse tree, combining several productions into one. For example, since no rules can be extracted for penalty-area, it is combined with its parent to form REGION → (left (penalty-area TEAM)), for which the pattern TEAM *left penalty area* is extracted.

The above algorithm is effective only when words linked to an MR predicate and its arguments stay close to each other, a property that we call *phrasal coherence*. Any links that destroy this property would lead to excessive node merging, a major cause of overfitting. Since building a model that strictly observes phrasal coherence often requires rules that model the reordering of tree nodes, our goal is to bootstrap the learning process by using a simpler, word-based alignment model that produces a generally coherent alignment, and then remove links that would cause excessive node merging before rule extraction takes place. Given an alignment, $\mathbf{a}$, we count the number of links that would prevent a rule from being extracted for each production in the MR parse. Then the total sum for all productions is obtained, denoted by $v(\mathbf{a})$. A greedy procedure is employed that repeatedly removes a link $a \in \mathbf{a}$ that would maximize $v(\mathbf{a}) - v(\mathbf{a} \backslash \{a\}) > 0$, until $v(\mathbf{a})$ cannot be further reduced. A link $w \leftrightarrow r$ is never removed if the translation probability, $\Pr(r|w)$, is greater than a certain threshold (0.9). To replenish the removed links, links from the most probable reverse alignment, $\tilde{\mathbf{a}}$ (obtained by treating the source language as target, and vice versa), are added to $\mathbf{a}$, as long as $\mathbf{a}$ remains n-to-1, and $v(\mathbf{a})$ is not increased.

5 Parameter Estimation

Once a lexicon is acquired, the next task is to learn a probabilistic model for the semantic parser. We propose a maximum-entropy model that defines a conditional probability distribution over derivations ($\mathbf{d}$) given the observed NL string ($\mathbf{e}$):

$$\Pr_\lambda(\mathbf{d}|\mathbf{e}) = \frac{1}{Z_\lambda(\mathbf{e})} \exp \sum_i \lambda_i f_i(\mathbf{d}) \qquad (2)$$

where f_i is a *feature function*, and $Z_\lambda(\mathbf{e})$ is a normalizing factor. For each rule r in the lexicon there is a feature function that returns the number of times r is used in a derivation. Also for each word w there is a feature function that returns the number of times w is generated from word gaps. Generation of unseen words is modeled using an extra feature whose value is the *total* number of words generated from word gaps. The number of features is quite modest (less than 3,000 in our experiments). A similar feature set is used by Zettlemoyer and Collins (2005).

Decoding of the model can be done in cubic time with respect to sentence length using the Viterbi algorithm. An Earley chart is used for keeping track of all derivations that are consistent with the input (Stolcke, 1995). The maximum conditional likelihood criterion is used for estimating the model parameters, λ_i. A Gaussian prior ($\sigma^2 = 1$) is used for regularizing the model (Chen and Rosenfeld, 1999). Since gold-standard derivations are not available in the training data, correct derivations must be treated as hidden variables. Here we use a version of im-

proved iterative scaling (IIS) coupled with EM (Riezler et al., 2000) for finding an optimal set of parameters.[1] Unlike the fully-supervised case, the conditional likelihood is not concave with respect to λ, so the estimation algorithm is sensitive to initial parameters. To assume as little as possible, λ is initialized to $\mathbf{0}$. The estimation algorithm requires statistics that depend on all possible derivations for a sentence or a sentence-MR pair. While it is not feasible to enumerate all derivations, a variant of the Inside-Outside algorithm can be used for efficiently collecting the required statistics (Miyao and Tsujii, 2002). Following Zettlemoyer and Collins (2005), only rules that are used in the best parses for the training set are retained in the final lexicon. All other rules are discarded. This heuristic, commonly known as *Viterbi approximation*, is used to improve accuracy, assuming that rules used in the best parses are the most accurate.

6 Experiments

We evaluated WASP in the ROBOCUP and GEOQUERY domains (see Section 2). To build a corpus for ROBOCUP, 300 pieces of coach advice were randomly selected from the log files of the 2003 ROBOCUP Coach Competition, which were manually translated into English (Kuhlmann et al., 2004). The average sentence length is 22.52. To build a corpus for GEOQUERY, 880 English questions were gathered from various sources, which were manually translated into the functional GEOQUERY language (Tang and Mooney, 2001). The average sentence length is 7.48, much shorter than ROBOCUP. 250 of the queries were also translated into Spanish, Japanese and Turkish, resulting in a smaller, multilingual data set.

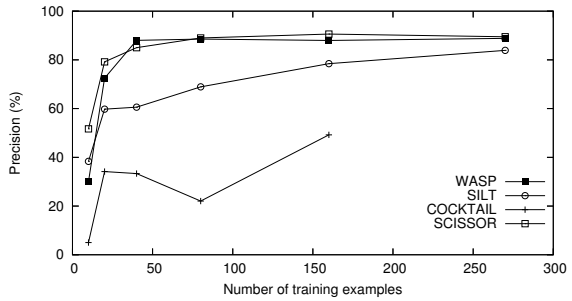
For each domain, there was a minimal set of *initial rules* representing knowledge needed for translating basic domain entities. These rules were always included in a lexicon. For example, in GEOQUERY, the initial rules were: NUM $\rightarrow \langle x, x \rangle$, for all $x \in \mathbb{R}$; CITY $\rightarrow \langle c,$ `cityid('c',_)` $\rangle$, for all city names c (e.g. *new york*); and similar rules for other types of names (e.g. rivers). Name translations were provided for the multilingual data set (e.g. CITY $\rightarrow \langle$*nyuu yooku*, `cityid('new york',_)` $\rangle$ for Japanese).

Standard 10-fold cross validation was used in our experiments. A semantic parser was learned from the training set. Then the learned parser was used to translate the test sentences into MRs. Translation failed when there were constructs that the parser did not cover. We counted the number of sentences that were translated into an MR, and the number of translations that were correct. For ROBOCUP, a translation was correct if it exactly matched the correct MR. For GEOQUERY, a translation was correct if it retrieved the same answer as the correct query. Using these counts, we measured the performance of the parser in terms of *precision* (percentage of translations that were correct) and *recall* (percentage of test sentences that were correctly translated). For ROBOCUP, it took 47 minutes to learn a parser using IIS. For GEOQUERY, it took 83 minutes.

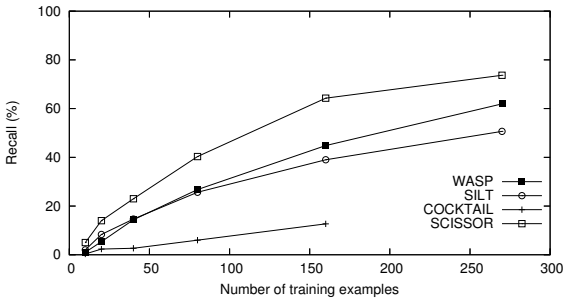
Figure 6 shows the performance of WASP compared to four other algorithms: SILT (Kate et al., 2005), COCKTAIL (Tang and Mooney, 2001), SCISSOR (Ge and Mooney, 2005) and Zettlemoyer and Collins (2005). Experimental results clearly show the advantage of extra supervision in SCISSOR and Zettlemoyer and Collins's parser (see Section 1). However, WASP performs quite favorably compared to SILT and COCKTAIL, which use the same training data. In particular, COCKTAIL, a deterministic shift-reduce parser based on inductive logic programming, fails to scale up to the ROBOCUP domain where sentences are much longer, and crashes on larger training sets due to memory overflow. WASP also outperforms SILT in terms of recall, where lexical learning is done by a local bottom-up search, which is much less effective than the word-alignment-based algorithm in WASP.

Figure 7 shows the performance of WASP on the multilingual GEOQUERY data set. The languages being considered differ in terms of word order: Subject-Verb-Object for English and Spanish, and Subject-Object-Verb for Japanese and Turkish. WASP's performance is consistent across these languages despite some slight differences, most probably due to factors other than word order (e.g. lower recall for Turkish due to a much larger vocabulary). Details can be found in a longer version of this paper (Wong, 2005).

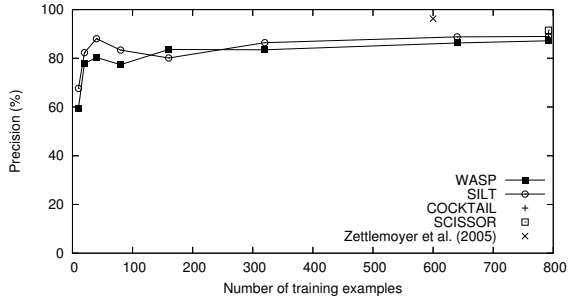
[1] We also implemented limited-memory BFGS (Nocedal, 1980). Preliminary experiments showed that it typically reduces training time by more than half with similar accuracy.



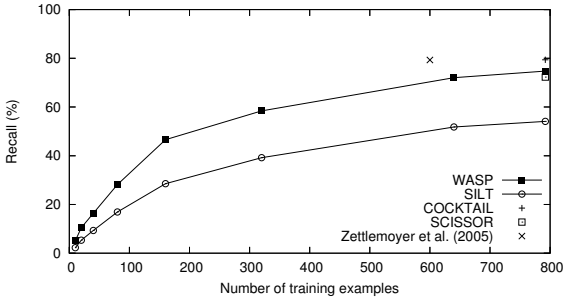
(a) Precision for ROBOCUP



(b) Recall for ROBOCUP

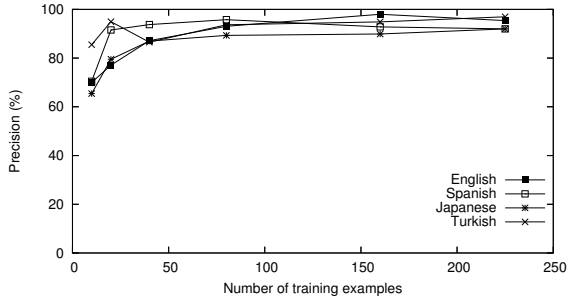


(c) Precision for GEOQUERY

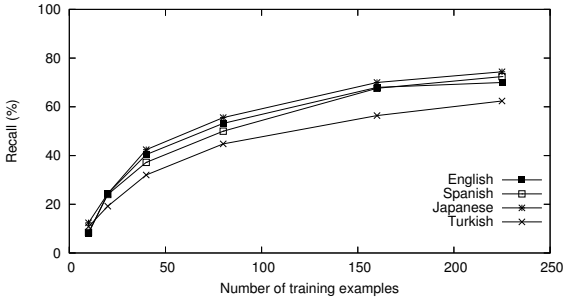


(d) Recall for GEOQUERY

Figure 6: Precision and recall learning curves comparing various semantic parsers



(a) Precision for GEOQUERY



(b) Recall for GEOQUERY

Figure 7: Precision and recall learning curves comparing various natural languages

7 Conclusion

We have presented a novel statistical approach to semantic parsing in which a word-based alignment model is used for lexical learning, and the parsing model itself can be seen as a syntax-based translation model. Our method is like many phrase-based translation models, which require a simpler, word-based alignment model for the acquisition of a phrasal lexicon (Och and Ney, 2003). It is also similar to the hierarchical phrase-based model of Chiang (2005), in which hierarchical phrase pairs, essentially SCFG rules, are learned through the use of a simpler, phrase-based alignment model. Our work shows that ideas from compiler theory (SCFG) and machine translation (word alignment models) can be successfully applied to semantic parsing, a closely-related task whose goal is to translate a natural language into a formal language.

Lexical learning requires word alignments that are phrasally coherent. We presented a simple greedy algorithm for removing links that destroy phrasal coherence. Although it is shown to be quite effective in the current domains, it is preferable to have a more principled way of *promoting* phrasal coherence. The problem is that, by treating MRL productions as atomic units, current word-based alignment models have *no* knowledge about the tree structure hidden in a linearized MR parse. In the future, we would like to develop a word-based alignment model that

is aware of the MRL syntax, so that better lexicons can be learned.

Acknowledgments

This research was supported by Defense Advanced Research Projects Agency under grant HR0011-04-1-0007.

References

A. V. Aho and J. D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*. Prentice Hall, Englewood Cliffs, NJ.

J. F. Allen. 1995. *Natural Language Understanding (2nd Ed.)*. Benjamin/Cummings, Menlo Park, CA.

I. Androutsopoulos, G. D. Ritchie, and P. Thanisch. 1995. Natural language interfaces to databases: An introduction. *Journal of Natural Language Engineering*, 1(1):29–81.

P. F. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312, June.

S. Chen and R. Rosenfeld. 1999. A Gaussian prior for smoothing maximum entropy models. Technical report, Carnegie Mellon University, Pittsburgh, PA.

M. Chen et al. 2003. Users manual: RoboCup soccer server manual for soccer server version 7.07 and later. Available at `http://sourceforge.net/projects/sserver/`.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. of ACL-05*, pages 263–270, Ann Arbor, MI, June.

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.

R. J. Kate, Y. W. Wong, and R. J. Mooney. 2005. Learning to transform natural to formal languages. In *Proc. of AAAI-05*, pages 1062–1068, Pittsburgh, PA, July.

G. Kuhlmann, P. Stone, R. J. Mooney, and J. W. Shavlik. 2004. Guiding a reinforcement learner with natural language advice: Initial results in RoboCup soccer. In *Proc. of the AAAI-04 Workshop on Supervisory Control of Learning and Adaptive Systems*, San Jose, CA, July.

K. Macherey, F. J. Och, and H. Ney. 2001. Natural language understanding using statistical machine translation. In *Proc. of EuroSpeech-01*, pages 2205–2208, Aalborg, Denmark.

I. D. Melamed. 2004. Statistical machine translation by parsing. In *Proc. of ACL-04*, pages 653–660, Barcelona, Spain.

S. Miller, D. Stallard, R. Bobrow, and R. Schwartz. 1996. A fully statistical approach to natural language interfaces. In *Proc. of ACL-96*, pages 55–61, Santa Cruz, CA.

Y. Miyao and J. Tsujii. 2002. Maximum entropy estimation for feature forests. In *Proc. of HLT-02*, San Diego, CA, March.

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

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.

S. Riezler, D. Prescher, J. Kuhn, and M. Johnson. 2000. Lexicalized stochastic modeling of constraint-based grammars using log-linear measures and EM training. In *Proc. of ACL-00*, pages 480–487, Hong Kong.

A. Stolcke. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201.

L. R. Tang and R. J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proc. of ECML-01*, pages 466–477, Freiburg, Germany.

Y. W. Wong. 2005. Learning for semantic parsing using statistical machine translation techniques. Technical Report UT-AI-05-323, Artificial Intelligence Lab, University of Texas at Austin, Austin, TX, October.

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.

ParaEval: Using Paraphrases to Evaluate Summaries Automatically

Liang Zhou, Chin-Yew Lin, Dragos Stefan Munteanu, and Eduard Hovy
University of Southern California
Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695
{liangz, cyl, dragos, hovy} @isi.edu

Abstract

ParaEval is an automated evaluation method for comparing reference and peer summaries. It facilitates a tiered-comparison strategy where recall-oriented global optimal and local greedy searches for paraphrase matching are enabled in the top tiers. We utilize a domain-independent paraphrase table extracted from a large bilingual parallel corpus using methods from Machine Translation (MT). We show that the quality of ParaEval's evaluations, measured by correlating with human judgments, closely resembles that of ROUGE's.

1 Introduction

Content coverage is commonly measured in summary comparison to assess how much information from the reference summary is included in a peer summary. Both manual and automatic methodologies have been used. Naturally, there is a great amount of confidence in manual evaluation since humans can infer, paraphrase, and use world knowledge to relate text units with similar meanings, but which are worded differently. Human efforts are preferred if the evaluation task is easily conducted and managed, and does not need to be performed repeatedly. However, when resources are limited, automated evaluation methods become more desirable.

For years, the summarization community has been actively seeking an automatic evaluation methodology that can be readily applied to various summarization tasks. ROUGE (Lin and Hovy, 2003) has gained popularity due to its simplicity and high correlation with human judgments. Even though validated by high correlations with human judgments gathered from previous Document Understanding Conference (DUC) experiments, current automatic procedures (Lin and Hovy, 2003; Hovy et al., 2005) only employ lexical *n-gram* matching. The lack of support for word or phrase matching that stretches beyond strict lexical matches has limited the expressiveness and utility of these methods. We need a mechanism that supplements literal matching—i.e. paraphrase and synonym—and approximates semantic closeness.

In this paper we present ParaEval, an automatic summarization evaluation method, which facilitates paraphrase matching in an overall three-level comparison strategy. At the top level, favoring higher coverage in reference, we perform an optimal search via dynamic programming to find multi-word to multi-word paraphrase matches between phrases in the reference summary (usually human-written) and those in the peer summary (system-generated). The non-matching fragments from the previous level are then searched by a greedy algorithm to find single-word paraphrase/synonym matches. At the third and the lowest level, we perform literal lexical unigram matching on the remaining texts. This tiered design for summary comparison guarantees at least a ROUGE-1 level of summary content matching if no paraphrases are found.

The first two levels employ a paraphrase table. Since manually created multi-word paraphrases—phrases determined by humans to be paraphrases of one another—are not available in sufficient quantities, we automatically build a paraphrase

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 447–454,
New York, June 2006.

table using methods from the Machine Translation (MT) field. The assumption made in creating this table is that if two English phrases are translated into the same foreign phrase with high probability (shown in the alignment results from a statistically trained alignment algorithm), then the two English phrases are paraphrases of each other.

This paper is organized in the following way: Section 2 introduces previous work in summarization evaluation; Section 3 describes the motivation behind this work; paraphrase acquisition is discussed in Section 4; Section 5 explains in detail our summary comparison mechanism; Section 6 validates ParaEval with human summary judgments; and we conclude and discuss future work in Section 7.

2 Previous Work

There has been considerable work in both manual and automatic summarization evaluations. Three most noticeable efforts in manual evaluation are SEE (Lin and Hovy, 2001), Factoid (Van Halteren and Teufel, 2003), and the Pyramid method (Nenkova and Passonneau, 2004).

SEE provides a user-friendly environment in which human assessors evaluate the quality of system-produced peer summary by comparing it to a reference summary. Summaries are represented by a list of summary units (sentences, clauses, etc.). Assessors can assign full or partial content coverage score to peer summary units in comparison to the corresponding reference summary units. Grammaticality can also be graded unit-wise.

The goal of the Factoid work is to compare the information content of different summaries of the same text and determine the minimum number of summaries, which was shown through experimentation to be 20-30, needed to achieve stable consensus among 50 human-written summaries.

The Pyramid method uses identified consensus—a pyramid of phrases created by annotators—from multiple reference summaries as the gold-standard reference summary. Summary comparisons are performed on Summarization Content Units (SCUs) that are approximately of clause length.

To facilitate fast summarization system design-evaluation cycles, ROUGE was created (Lin and Hovy, 2003). It is an automatic evaluation package that measures a number of *n-gram* co-occurrence statistics between peer and reference summary pairs. ROUGE was inspired by BLEU (Papineni et al., 2001) which was adopted by the machine translation (MT) community for automatic MT evaluation. A problem with ROUGE is that the summary units used in automatic comparison are of fixed length. A more desirable design is to have summary units of variable size. This idea was implemented in the Basic Elements (BE) framework (Hovy et al., 2005) which has not been completed due to its lack of support for paraphrase matching. Both ROUGE and BE have been shown to correlate well with past DUC human summary judgments, despite incorporating only lexical matching on summary units (Lin and Hovy, 2003; Hovy et al., 2005).

SCU1: the crime in question was the Lockerbie {Scotland} bombing
1 [for the Lockerbie bombing]
2 [for blowing up] [over Lockerbie, Scotland]
3 [of bombing] [over Lockerbie, Scotland]
4 [was blown up over Lockerbie, Scotland,]
5 [the bombing of Pan Am Flight 103]
6 [bombing over Lockerbie, Scotland,]
7 [for Lockerbie bombing]
8 [bombing of Pan Am flight 103 over Lockerbie.]
9 [linked to the Lockerbie bombing]
10 [in the Lockerbie bombing case.]

Figure 1. Paraphrases created by Pyramid annotation.

3 Motivation

3.1 Paraphrase Matching

An important difference that separates current manual evaluation methods from their automatic counterparts is that semantic matching of content units is performed by human summary assessors. An essential part of the semantic matching involves paraphrase matching—determining whether phrases worded differently carry the same semantic information. This paraphrase matching process is observed in the Pyramid annotation procedure shown in (Nenkova and Passonneau, 2004) over three summary sets (10 summaries each). In the example shown in Figure 1 (reproduced from Pyramid results), each of the 10 phrases (numbered 1 to 10) extracted from summary sentences carries the same *semantic content* as the overall summary content unit labeled SCU1 does. Each extracted phrase is identified as a summary content unit (SCU). In our work in building an automatic evaluation procedure that enables paraphrase

matching, we aim to automatically identify these 10 phrases as paraphrases of one another.

3.2 Synonymy Relations

Synonym matching and paraphrase matching are often mentioned in the same context in discussions of extending current automated summarization evaluation methods to incorporate the matching of semantic units. While evaluating automatically extracted paraphrases via WordNet (Miller et al., 1990), Barzilay and McKeown (2001) quantitatively validated that synonymy is not the only source of paraphrasing. We envisage that this claim is also valid for summary comparisons.

From an in-depth analysis on the manually created SCUs of the DUC2003 summary set D30042 (Nenkova and Passonneau, 2004), we find that 54.48% of 1746 cases where a non-stop word from one SCU did not match with its supposedly human-aligned pairing SCUs are in need of some level of paraphrase matching support. For example, in the first two extracted SCUs (labeled as 1 and 2) in Figure 1—"for the Lockerbie bombing" and "for blowing up … over Lockerbie, Scotland"—no non-stop word other than the word "Lockerbie" occurs in both phrases. But these two phrases were judged to carry the same semantic meaning because human annotators think the word "bombing" and the phrase "blowing up" refer to the same action, namely the one associated with "explosion." However, "bombing" and "blowing up" cannot be matched through synonymy relations by using WordNet, since one is a noun and the other is a verb phrase (if tagged within context). Even when the search is extended to finding synonyms and hypernyms for their categorical variants and/or using other parts of speech (verb for "bombing" and noun phrase for "blowing up"), a match still cannot be found.

To include paraphrase matching in summary evaluation, a collection of less-strict paraphrases must be created and a matching strategy needs to be investigated.

4 Paraphrase Acquisition

Paraphrases are alternative verbalizations for conveying the same information and are required by many Natural Language Processing (NLP) applications. In particular, summary creation and evaluation methods need to recognize paraphrases and their semantic equivalence. Unfortunately, we have yet to incorporate into the evaluation framework previous findings in paraphrase identification and extraction (Barzilay and McKeown, 2001; Pang et al., 2003; Bannard and Callison-Burch, 2005).

4.1 Related Work on Paraphrasing

Three major approaches in paraphrase collection are manual collection (domain-specific), collection utilizing existing lexical resources (i.e. WordNet), and derivation from corpora. Hermjakob et al. (2002) view paraphrase recognition as reformulation by pattern recognition. Pang et al. (2003) use word lattices as paraphrase representations from semantically equivalent translations sets. Using parallel corpora, Barzilay and McKeown (2001) identify paraphrases from multiple translations of classical novels, where as Bannard and Callison-Burch (2005) develop a probabilistic representation for paraphrases extracted from large Machine Translation (MT) data sets.

4.2 Extracting Paraphrases

Our method to automatically construct a large domain-independent paraphrase collection is based on the assumption that two different English phrases of the same meaning may have the same translation in a foreign language.

Phrase-based Statistical Machine Translation (SMT) systems analyze large quantities of bilingual parallel texts in order to learn translational alignments between pairs of words and phrases in two languages (Och and Ney, 2004). The sentence-based translation model makes word/phrase alignment decisions probabilistically by computing the optimal model parameters with application of the statistical estimation theory. This alignment process results in a corpus of word/phrase-aligned parallel sentences from which we can extract phrase pairs that are translations of each other. We ran the alignment algorithm from (Och and Ney, 2003) on a Chinese-English parallel corpus of 218 million English words. Phrase pairs are extracted by following the method described in (Och and Ney, 2004) where all contiguous phrase pairs having consistent alignments are extraction candidates. The resulting phrase table is of high quality; both the alignment models and phrase extraction meth-

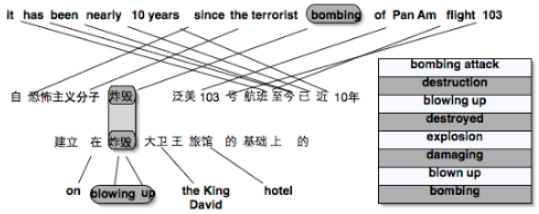


Figure 2. An example of paraphrase extraction.

ods have been shown to produce very good results for SMT. Using these pairs we build paraphrase sets by joining together all English phrases with the same Chinese translation. Figure 2 shows an example word/phrase alignment for two parallel sentence pairs from our corpus where the phrases "blowing up" and "bombing" have the same Chinese translation. On the right side of the figure we show the paraphrase set which contains these two phrases, which is typical in our collection of extracted paraphrases.

5 Summary Comparison in ParaEval

This section describes the process of comparing a peer summary against a reference summary and the summary grading mechanism.

5.1 Description

We adopt a three-tier matching strategy for summary comparison. The score received by a peer summary is the ratio of the number of reference words matched to the total number of words in the reference summary. The total number of matched reference words is the sum of matched words in reference throughout all three tiers. At the top level, favoring high recall coverage, we perform an optimal search to find multi-word paraphrase matches between phrases in the reference summary and those in the peer. Then a greedy search is performed to find single-word paraphrase/synonym matches among the remaining text. Operations conducted in these two top levels are marked as linked rounded rectangles in Figure 3. At the bottom level, we find lexical identity matches, as marked in rectangles in the example. If no paraphrases are found, this last level provides a guarantee of lexical comparison that is equivalent to what other automated systems give. In our system, the bottom level currently performs unigram matching. Thus, we are ensured with at least a ROUGE-1 type of summary comparison. Alternatively, equivalence of other ROUGE configurations can replace the ROUGE-1 implementation.

There is no theoretical reason why the first two levels should not merge. But due to high computational cost in modeling an optimal search, the separation is needed. We explain this in detail below.

5.2 Multi-Word Paraphrase Matching

In this section we describe the algorithm that performs the multi-word paraphrase matching between phrases from reference and peer summaries. Using the example in Figure 3, this algorithm creates the phrases shown in the rounded rectangles and establishes the appropriate links indicating corresponding paraphrase matches.

Problem Description

Measuring content coverage of a peer summary using a single reference summary requires computing the recall score of how much information from the reference summary is included in the peer. A summary unit, either from reference or peer, cannot be matched for more than once. For

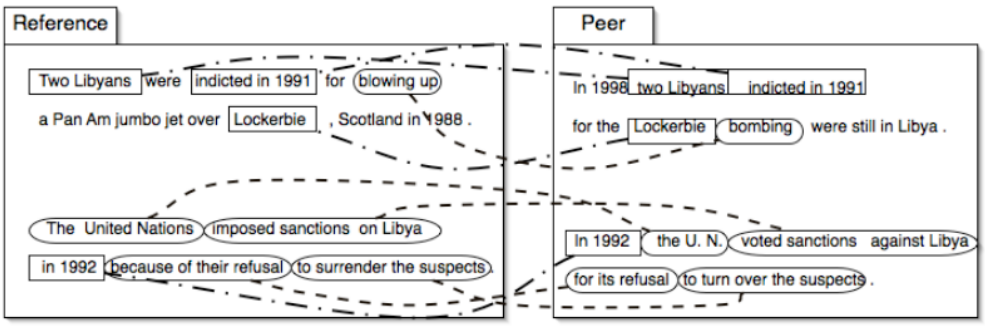


Figure 3. Comparison of summaries.

example, the phrase "imposed sanctions on Libya" (r_1) in Figure 3's reference summary was matched with the peer summary's "voted sanctions against Libya" (p_1). If later in the peer summary there is another phrase p_2 that is also a paraphrase of r_1, the match of r_1 cannot be counted twice. Conversely, double counting is not permissible for phrase/words in the peer summary, either.

We conceptualize the comparison of peer against reference as a task that is to complete over several time intervals. If the reference summary contains n sentences, there will be n time intervals, where at time t_i, phrases from a particular sentence i of the reference summary are being considered with all possible phrases from the peer summary for paraphrase matches. A decision needs to be made at each time interval:

- Do we employ a local greedy match algorithm that is recall generous (preferring more matched words from reference) towards only the reference sentence currently being analyzed,
- Or do we need to explore globally, inspecting all reference sentences and find the best overall matching combinations?

Consider the scenario in Figure 4:

1) at t_0: $L(p_1 = r_2) > L(p_2 = r_1)$ and r_2 contains r_1. A local search algorithm leads to $match(p_1, r_2)$. $L()$ indicates the number of words in reference matched by the peer phrase through paraphrase matching and $match()$ indicates a paraphrase match has occurred (more in the figure).

2) at t_1: $L(p_1 = r_3) > L(p_1 = r_2)$. A global algorithm reverses the decision $match(p_1, r_2)$ made at t_0 and concludes $match(p_1, r_3)$ and $match(p_2, r_1)$. A local search algorithm would have returned no match.

Clearly, the global search algorithm achieves higher overall recall (in words). The matching of paraphrases between a reference and its peer becomes a global optimization problem, maximizing the content coverage of the peer compared in reference.

Solution Model

We use dynamic programming to derive the solution of finding the best paraphrase-matching combinations. The optimization problem is as follows: Sentences from a reference summary and a peer summary can be broken into phrases of various lengths. A paraphrase lookup table is used to find whether a reference phrase and a peer phrase are paraphrases of each other. What is the optimal paraphrase matching combination of phrases from reference and peer that gives the highest recall score (in number of matched reference words) for this given peer? The solution should be recall oriented (favoring a peer phrase that matches more reference words than those match less).

Following (Trick, 1997), the solution can be characterized as:

1) This problem can be divided into n stages corresponding to the n sentences of the reference summary. At each stage, a decision is required to determine the best combination of matched paraphrases between the reference sentence and the entire peer summary that results in no double counting of phrases on the peer side. There is no double counting of reference phrases across stages since we are processing one reference sentence at a time and are finding the best paraphrase matches using the entire peer summary. As long as there is no double counting in peers, we are guaranteed to have none in reference, either.

2) At each stage, we define a number of possible states as follows. If, out of all possible phrases of any length extracted from the reference sentence, m phrases were found to have matching paraphrases in the peer summary, then a state is any subset of the m phrases.

3) Since no double counting in matched phrases/words is allowed in either the reference summary or the peer summary, the decision of which phrases (leftover text segments in reference

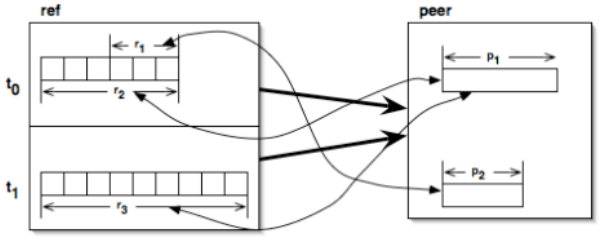


Figure 4. Local vs. global paraphrase matching.

P_j and r_i represent phrases chosen for paraphrase matching from peer and reference respectively.

$P_j = r_i$ indicates that the phrase P_j from peer is found to be a paraphrase to the phrase r_i from reference.

$L(P_j = r_i)$ indicates the number of words matched by P_j in r_i when they are found to be paraphrases of each other.

$L(P_j = r_i)$ and $L(P_j = r_{i+1})$ may not be equal if the number of words in r_i, indicated by $L(r_i)$, does not equal to the number of words in r_{i+1}, indicated by $L(r_{i+1})$.

and in peer) are allowed to match for the next stage is made in the current stage.

4) *Principle of optimality*: at a given state, it is not necessary to know what matches occurred at previous stages, only on the accumulated recall score (matched reference words) from previous stages and what text segments (phrases) in peer have not been taken/matched in previous stages.

5) There exists a recursive relationship that identifies the optimal decision for stage s (out of n total stages), given that stage $s+1$ has already been solved.

6) The final stage, n (last sentence in reference), is solved by choosing the state that has the highest accumulated recall score and yet resulted no double counting in any phrase/word in peer the summary.

Figure 5 demonstrates the optimal solution (12 reference words matched) for the example shown in Figure 4. We can express the calculations in the following formulas:

$$f_1(x_s) = \max_{x_b : c(x_b)} \{r(x_s)\}$$

and

$$f_y(x_b) = \max_{x_b : c(x_b)} \{r(x_b) + f_{y-1}(x_b - c(x_b))\}$$

where $f_y(x_b)$ denotes the optimal recall coverage (number of words in the reference summary matched by the phrases from the peer summary) at state x_b in stage y. $r(x_b)$ is the recall coverage given state x_b. And $c(x_b)$ records the phrases matched in peer with no double counting, given state x_b.

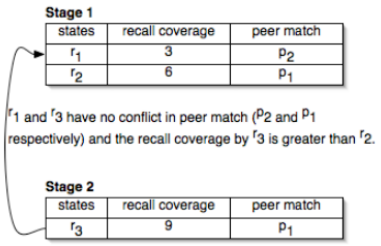


Figure 5. Solution for the example in Figure 4.

5.3 Synonym Matching

All paraphrases whose pairings do not involve multi-word to multi-word matching are called synonyms in our experiment. Since these phrases have either a *n-to-1* or *1-to-n* matching ratio (such as the phrases "blowing up" and "bombing"), a greedy algorithm favoring higher recall coverage reduces the state creation and stage comparison costs associated with the optimal procedure ($\mathbf{O}(m^6)$: $\mathbf{O}(m^3)$ for state creation, and for 2 stages at any time)). The paraphrase table described in Section 4 is used.

Synonym matching is performed only on parts of the reference and peer summaries that were not matched from the multi-word paraphrase-matching phase.

5.4 Lexical Matching

This matching phase performs straightforward lexical matching, as exemplified by the text fragments marked in rectangles in Figure 3. Unigrams are used as the units for counting matches in accordance with the previous two matching phases.

During all three matching phases, we employed a ROUGE-1 style of counting. Other alternatives, such as ROUGE-2, ROUGE-SU4, etc., can easily be adapted to each phase.

6 Evaluation of ParaEval

To evaluate and validate the effectiveness of an automatic evaluation metric, it is necessary to show that automatic evaluations correlate with human assessments highly, positively, and consistently (Lin and Hovy, 2003). In other words, an automatic evaluation procedure should be able to distinguish good and bad summarization systems by assigning scores with close resemblance to humans' assessments.

6.1 Document Understanding Conference

The Document Understanding Conference has provided large-scale evaluations on both human-created and system-generated summaries annually. Research teams are invited to participate in solving summarization problems with their systems. System-generated summaries are then assessed by humans and/or automatic evaluation procedures. The collection of human judgments on systems and their summaries has provided a test-bed for developing and validating automated summary grading methods (Lin and Hovy, 2003; Hovy et al., 2005).

The correlations reported by ROUGE and BE show that the evaluation correlations between these two systems and DUC human evaluations are much higher on single-document summarization tasks. One possible explanation is that when sum-

marizing from only one source (text), both human- and system-generated summaries are mostly extractive. The reason for humans to take phrases (or maybe even sentences) verbatim is that there is less motivation to abstract when the input is not highly redundant, in contrast to input for multi-document summarization tasks, which we speculate allows more abstracting. ROUGE and BE both facilitate lexical *n-gram* matching, hence, achieving amazing correlations. Since our baseline matching strategy is lexically based when paraphrase matching is not activated, validation on single-doc summarization results is not repeated in our experiment.

6.2 Validation and Discussion

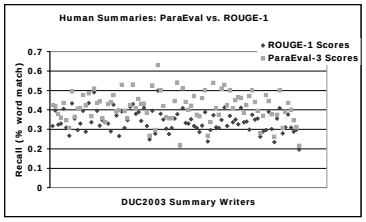
We use summary judgments from DUC2003's multi-document summarization (MDS) task to evaluate ParaEval. During DUC2003, participating systems created short summaries (~100 words) for 30 document sets. For each set, one assessor-written summary was used as the reference to compare peer summaries created by 18 automatic systems (including baselines) and 3 other human-written summaries. A system ranking was produced by taking the averaged performance on all summaries created by systems. This evaluation process is replicated in our validation setup for ParaEval. In all, 630 summary pairs were compared. Pearson's correlation coefficient is computed for the validation tests, using DUC2003 assessors' results as the gold standard.

Table 1 illustrates the correlation figures from the DUC2003 test set. ParaEval-para_only shows the correlation result when using only paraphrase and synonym matching, without the baseline unigram matching. ParaEval-2 uses multi-word paraphrase matching and unigram matching, omitting the greedy synonym-matching phrase. ParaEval-3 incorporates matching at all three granularity levels.

We see that the current implementation of ParaEval closely resembles the way ROUGE-1 differentiates system-generated summaries. We believe this is due to the identical calculations of recall scores. The score that a peer summary receives from ParaEval depends on the number of words matched in the reference summary from its paraphrase, synonym, and unigram matches. The counting of individual words in reference indicates a ROUGE-1 design in grading. However, a detailed examination on individual reference-peer comparisons shows that paraphrase and synonym comparisons and matches, in addition to lexical *n-gram* matching, do measure a higher level of content coverage. This is demonstrated in Figure 6a and b. Strict unigram matching reflects the content retained by a peer summary mostly in the 0.2-0.4 ranges in recall, shown as dark-colored dots in the graphs. Allowing paraphrase and synonym matching increases the detection of peer coverage to the range of 0.3-0.5, shown as light-colored dots.

DUC-2003	Pearson
ROUGE-1	0.622
ParaEval-para_only	0.41
ParaEval-2	0.651
ParaEval-3	0.657

Table 1. Correlation with DUC 2003 MDS results.



a). Human-written summaries.

b). System-generated summaries.

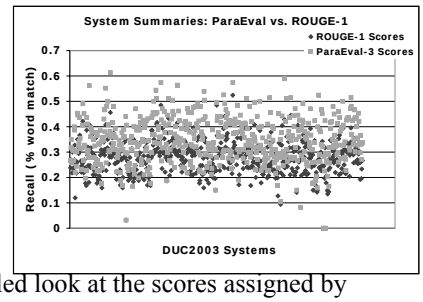


Figure 6. A detailed look at the scores assigned by lexical and paraphrase/synonym comparisons.

We conducted a manual evaluation to further examine the paraphrases being matched. Using 10 summaries from the Pyramid data, we asked three human subjects to judge the validity of 128 (randomly selected) paraphrase pairs extracted and identified by ParaEval. Each pair of paraphrases was coupled with its respective sentences as contexts. All paraphrases judged were multi-word. ParaEval received an average precision of 68.0%. The complete agreement between judges is 0.582 according to the Kappa coefficient (Cohen, 1960). In Figure 7, we show two examples that the human judges consider to be good paraphrases produced and matched by ParaEval. Judges voiced difficul-

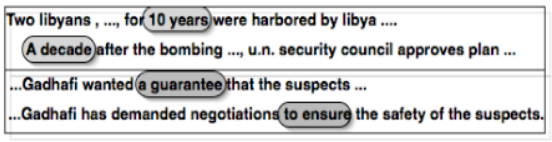


Figure 7. Paraphrases matched by ParaEval.

ties in determining "semantic equivalence." There were cases where paraphrases would be generally interchangeable but could not be matched because of non-semantic equivalence in their contexts. And there were paraphrases that were determined as matches, but if taken out of context, would not be direct replacements of each other. These two situations are where the judges mostly disagreed.

7 Conclusion and Future Work

In this paper, we have described an automatic summarization evaluation method, ParaEval, that facilitates paraphrase matching using a large domain-independent paraphrase table extracted from a bilingual parallel corpus. The three-layer matching strategy guarantees a ROUGE-like baseline comparison if paraphrase matching fails.

The paraphrase extraction module from the current implementation of ParaEval does not discriminate among the phrases that are found to be paraphrases of one another. We wish to incorporate the probabilistic paraphrase extraction model from (Bannard and Callison-Burch, 2005) to better approximate the relations between paraphrases. This adaptation will also lead to a stochastic model for the low-level lexical matching and scoring.

We chose English-Chinese MT parallel data because they are news-oriented which coincides with the task genre from DUC. However, it is unknown how large a parallel corpus is sufficient in providing a paraphrase collection good enough to help the evaluation process. The quality of the paraphrase table is also affected by changes in the domain and language pair of the MT parallel data. We plan to use ParaEval to investigate the impact of these changes on paraphrase quality under the assumption that better paraphrase collections lead to better summary evaluation results.

The immediate impact and continuation of the described work would be to incorporate paraphrase matching and extraction into the summary creation process. And with ParaEval, it is possible for us to evaluate systems that do incorporate some level of abstraction, especially paraphrasing.

References

Bannard, C. and C. Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. *Proceedings of ACL-2005*.

Barzilay, R. and K. McKeown. 2001. Extracting paraphrases from a parallel corpus. Proceedings of *ACL/EACL-2001*.

Brown, P. F., S. A. Della Pietra, V. J. Della Pietra, R. L. Mercer. 1993. The mathematics of machine translation: Parameter estimation. *Computational Linguistics*, 19(2): 263—311, 1993.

Cohen, J. 1960. A coefficient of agreement for nominal scales. *Education and Psychological Measurement*, 43(6):37—46.

Diab, M. and P. Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. *Proceedings of ACL-2002*.

DUC. 2001—2005. Document Understanding Conferences.

Hermjakob, U., A. Echihabi, and D. Marcu. 2002. Natural language based reformulation resource and web exploitation for question answering. *Proceedings of TREC-2002*.

Hovy, E, C.Y. Lin, and L. Zhou. 2005. Evaluating DUC 2005 using basic elements. *Proceedings of DUC-2005*.

Hovy, E., C.Y. Lin, L. Zhou, and J. Fukumoto. 2005a. Basic Elements. http://www.isi.edu/~cyl/BE.

Lin, C.Y. 2001. http://www.isi.edu/~cyl/SEE.

Lin, C.Y. and E. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. *Proceedings of the HLT-2003*.

Miller, G.A., R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. 1990. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4): 235—245.

Nenkova, A. and R. Passonneau. 2004. Evaluating content selection in summarization: the pyramid method. *Proceedings of the HLT-NAACL 2004*.

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

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

Pang, B. , K. Knight and D. Marcu. 2003. Syntax-based alignment of multiple translations: extracting paraphrases and generating new sentences. *Proceedings of HLT/NAACL-2003*.

Papineni, K., S. Roukos, T. Ward, and W. J. Zhu. IBM research report Bleu: a method for automatic evaluation of machine translation *IBM Research Division Technical Report*, RC22176, 2001.

Trick, M. A. 1997. A tutorial on dynamic programming.http://mat.gsia.cmu.edu/classes/dynamic/dynamic.html.

Van Halteren, H. and S. Teufel. 2003. Examining the consensus between human summaries: initial experiments with factoid analysis. *Proceedings of HLT-2003*.

Paraphrasing for Automatic Evaluation

David Kauchak
Department of Computer Science
University of California, San Diego
dkauchak@cs.ucsd.edu

Regina Barzilay
CSAIL
Massachusetts Institute of Technology
regina@csail.mit.edu

Abstract

This paper studies the impact of paraphrases on the accuracy of automatic evaluation. Given a reference sentence and a machine-generated sentence, we seek to find a paraphrase of the reference sentence that is closer in wording to the machine output than the original reference. We apply our paraphrasing method in the context of machine translation evaluation. Our experiments show that the use of a paraphrased synthetic reference refines the accuracy of automatic evaluation. We also found a strong connection between the quality of automatic paraphrases as judged by humans and their contribution to automatic evaluation.

1 Introduction

The use of automatic methods for evaluating machine-generated text is quickly becoming mainstream in natural language processing. The most notable examples in this category include measures such as BLEU and ROUGE which drive research in the machine translation and text summarization communities. These methods assess the quality of a machine-generated output by considering its similarity to a reference text written by a human. Ideally, the similarity would reflect the semantic proximity between the two. In practice, this comparison breaks down to n-gram overlap between the reference and the machine output.

1a. However, Israel's reply failed to completely clear the U.S. suspicions.
1b. However, Israeli answer unable to fully remove the doubts.

Table 1: A reference sentence and corresponding machine translation from the NIST 2004 MT evaluation.

Consider the human-written translation and the machine translation of the same Chinese sentence shown in Table 1. While the two translations convey the same meaning, they share only auxiliary words. Clearly, any measure based on word overlap will penalize a system for generating such a sentence. The question is whether such cases are common phenomena or infrequent exceptions. Empirical evidence supports the former. Analyzing 10,728 reference translation pairs[1] used in the NIST 2004 machine translation evaluation, we found that only 21 (less than 0.2%) of them are identical. Moreover, 60% of the pairs differ in at least 11 words. These statistics suggest that without accounting for paraphrases, automatic evaluation measures may never reach the accuracy of human evaluation.

As a solution to this problem, researchers use multiple references to refine automatic evaluation. Papineni et al. (2002) shows that expanding the number of references reduces the gap between automatic and human evaluation. However, very few human annotated sets are augmented with multiple references and those that are available are relatively

[1]Each pair included different translations of the same sentence, produced by two human translators.

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 455–462,
New York, June 2006.

small in size. Moreover, access to several references does not guarantee that the references will include the same words that appear in machine-generated sentences.

In this paper, we explore the use of paraphrasing methods for refinement of automatic evaluation techniques. Given a reference sentence and a machine-generated sentence, we seek to find a paraphrase of the reference sentence that is closer in wording to the machine output than the original reference. For instance, given the pair of sentences in Table 1, we automatically transform the reference sentence (1a.) into

> However, Israel's *answer* failed to completely *remove* the U.S. suspicions.

Thus, among many possible paraphrases of the reference, we are interested only in those that use words appearing in the system output. Our paraphrasing algorithm is based on the *substitute in context* strategy. First, the algorithm identifies pairs of words from the reference and the system output that could potentially form paraphrases. We select these candidates using existing lexico-semantic resources such as WordNet. Next, the algorithm tests whether the candidate paraphrase is admissible in the context of the reference sentence. Since even synonyms cannot be substituted in any context (Edmonds and Hirst, 2002), this filtering step is necessary. We predict whether a word is appropriate in a new context by analyzing its distributional properties in a large body of text. Finally, paraphrases that pass the filtering stage are used to rewrite the reference sentence.

We apply our paraphrasing method in the context of machine translation evaluation. Using this strategy, we generate a new sentence for every pair of human and machine translated sentences. This synthetic reference then replaces the original human reference in automatic evaluation.

The key findings of our work are as follows:
(1) **Automatically generated paraphrases improve the accuracy of the automatic evaluation methods.** Our experiments show that evaluation based on paraphrased references gives a better approximation of human judgments than evaluation that uses original references.

(2) **The quality of automatic paraphrases determines their contribution to automatic evaluation.** By analyzing several paraphrasing resources, we found that the accuracy and coverage of a paraphrasing method correlate with its utility for automatic MT evaluation.

Our results suggest that researchers may find it useful to augment standard measures such as BLEU and ROUGE with paraphrasing information thereby taking more semantic knowledge into account.

In the following section, we provide an overview of existing work on automatic paraphrasing. We then describe our paraphrasing algorithm and explain how it can be used in an automatic evaluation setting. Next, we present our experimental framework and data and conclude by presenting and discussing our results.

2 Related Work

Automatic Paraphrasing and Entailment Our work is closely related to research in automatic paraphrasing, in particular, to sentence level paraphrasing (Barzilay and Lee, 2003; Pang et al., 2003; Quirk et al., 2004). Most of these approaches learn paraphrases from a parallel or comparable monolingual corpora. Instances of such corpora include multiple English translations of the same source text written in a foreign language, and different news articles about the same event. For example, Pang et al. (2003) expand a set of reference translations using syntactic alignment, and generate new reference sentences that could be used in automatic evaluation.

Our approach differs from traditional work on automatic paraphrasing in goal and methodology. Unlike previous approaches, we are not aiming to produce *any* paraphrase of a given sentence since paraphrases induced from a parallel corpus do not necessarily produce a rewriting that makes a reference closer to the system output. Thus, we focus on words that appear in the system output and aim to determine whether they can be used to rewrite a reference sentence.

Our work also has interesting connections with research on automatic textual entailment (Dagan et al., 2005), where the goal is to determine whether a given sentence can be inferred from text. While we are not assessing an inference relation between a reference and a system output, the two tasks face similar challenges. Methods for entailment

recognition extensively rely on lexico-semantic resources (Haghighi et al., 2005; Harabagiu et al., 2001), and we believe that our method for contextual substitution can be beneficial in that context.

Automatic Evaluation Measures A variety of automatic evaluation methods have been recently proposed in the machine translation community (NIST, 2002; Melamed et al., 2003; Papineni et al., 2002). All these metrics compute n-gram overlap between a reference and a system output, but measure the overlap in different ways. Our method for reference paraphrasing can be combined with any of these metrics. In this paper, we report experiments with BLEU due to its wide use in the machine translation community.

Recently, researchers have explored additional knowledge sources that could enhance automatic evaluation. Examples of such knowledge sources include stemming and TF-IDF weighting (Babych and Hartley, 2004; Banerjee and Lavie, 2005). Our work complements these approaches: we focus on the impact of paraphrases, and study their contribution to the accuracy of automatic evaluation.

3 Methods

The input to our method consists of a reference sentence $R = r_1 \dots r_m$ and a system-generated sentence $W = w_1 \dots w_p$ whose words form the sets $\mathcal{R}$ and $\mathcal{W}$ respectively. The output of the model is a synthetic reference sentence S_{RW} that preserves the meaning of R and has maximal word overlap with W. We generate such a sentence by substituting words from R with contextually equivalent words from W.

Our algorithm first selects pairs of candidate word paraphrases, and then checks the likelihood of their substitution in the context of the reference sentence.

Candidate Selection We assume that words from the reference sentence that already occur in the system generated sentence should not be considered for substitution. Therefore, we focus on unmatched pairs of the form $\{(r, w)|r \in \mathcal{R} - \mathcal{W}, w \in \mathcal{W} - \mathcal{R}\}$. From this pool, we select candidate pairs whose members exhibit high semantic proximity. In our experiments we compute semantic similarity using WordNet, a large-scale lexico-semantic resource employed in many NLP applications for similar purposes. We consider a pair as a substitution candidate if its members are synonyms in WordNet.

2a. It is **hard** to believe that such tremendous changes have taken **place** for those people and lands that I have never stopped missing while living abroad.
2b. For someone born here but has been sentimentally attached to a foreign country far from **home**, it is **difficult** to believe this kind of changes.

Table 2: A reference sentence and a corresponding machine translation. Candidate paraphrases are in bold.

Applying this step to the two sentences in Table 2, we obtain two candidate pairs (**home**, **place**) and (**difficult**, **hard**).

Contextual Substitution The next step is to determine for each candidate pair (r_i, w_j) whether w_j is a valid substitution for r_i in the context of $r_1 \dots r_{i-1} \square r_{i+1} \dots r_m$. This filtering step is essential because synonyms are not universally substitutable[2]. Consider the candidate pair (**home**, **place**) from our example (see Table 2). Words **home** and **place** are paraphrases in the sense of "habitat", but in the reference sentence "**place**" occurs in a different sense, being part of the collocation "take place". In this case, the pair (**home**, **place**) cannot be used to rewrite the reference sentence.

We formulate contextual substitution as a binary classification task: given a context $r_1 \dots r_{i-1} \square r_{i+1} \dots r_m$, we aim to predict whether w_j can occur in this context at position i. For each candidate word w_j we train a classifier that models contextual preferences of w_j. To train such a classifier, we collect a large corpus of sentences that contain the word w_j and an equal number of randomly extracted sentences that do not contain this word. The former category forms positive instances, while the latter represents the negative. For the negative examples, a random position in a sentence is selected for extracting the context. This corpus is acquired automatically, and does not require any manual annotations.

[2]This can explain why previous attempts to use WordNet for generating sentence-level paraphrases (Barzilay and Lee, 2003; Quirk et al., 2004) were unsuccessful.

We represent context by n-grams and local collocations, features typically used in supervised word sense disambiguation. Both n-grams and collocations exclude the word w_j. An n-gram is a sequence of n adjacent words appearing in $r_1 \dots r_{i-1} \square r_{i+1} \dots r_m$. A local collocation also takes into account the position of an n-gram with respect to the target word. To compute local collocations for a word at position i, we extract all n-grams ($n = 1 \dots 4$) beginning at position $i - 2$ and ending at position $i + 2$. To make these position dependent, we prepend each of them with the length and starting position.

Once the classifier[3] for w_j is trained, we apply it to the context $r_1 \dots r_{i-1} \square r_{i+1} \dots r_m$. For positive predictions, we rewrite the string as $r_1 \dots r_{i-1} w_j r_{i+1} \dots r_m$. In this formulation, all substitutions are tested independently.

For the example from Table 2, only the pair (**difficult**, **hard**) passes this filter, and thus the system produces the following synthetic reference:

> For someone born here but has been sentimentally attached to a foreign country far from home, it is **hard** to believe this kind of changes.

The synthetic reference keeps the meaning of the original reference, but has a higher word overlap with the system output.

One of the implications of this design is the need to develop a large number of classifiers to test contextual substitutions. For each word to be inserted into a reference sentence, we need to train a separate classifier. In practice, this requirement is not a significant burden. The training is done off-line and only once, and testing for contextual substitution is instantaneous. Moreover, the first filtering step effectively reduces the number of potential candidates. For example, to apply this approach to the 71,520 sentence pairs from the MT evaluation set (described in Section 4.1.2), we had to train 2,380 classifiers.

We also discovered that the key to the success of this approach is the size of the corpus used for training contextual classifiers. We derived training corpora from the English Gigaword corpus, and the average size of a corpus for one classifier is 255,000 sentences. We do not attempt to substitute any words that have less that 10,000 appearances in the Gigaword corpus.

[3] In our experiments, we used the publicly available BoosTexter classifier (Schapire and Singer, 2000) for this task.

4 Experiments

Our primary goal is to investigate the impact of machine-generated paraphrases on the accuracy of automatic evaluation. We focus on automatic evaluation of machine translation due to the availability of human annotated data in that domain. The hypothesis is that by using a synthetic reference translation, automatic measures approximate better human evaluation. In section 4.2, we test this hypothesis by comparing the performance of BLEU scores with and without synthetic references.

Our secondary goal is to study the relationship between the quality of paraphrases and their contribution to the performance of automatic machine translation evaluation. In section 4.3, we present a manual evaluation of several paraphrasing methods and show a close connection between intrinsic and extrinsic assessments of these methods.

4.1 Experimental Set-Up

We begin by describing relevant background information, including the BLEU evaluation method, the test data set, and the alternative paraphrasing methods considered in our experiments.

4.1.1 BLEU

BLEU is the basic evaluation measure that we use in our experiments. It is the geometric average of the n-gram precisions of candidate sentences with respect to the corresponding reference sentences, times a brevity penalty. The BLEU score is computed as follows:

$$\begin{aligned} BLEU &= BP \cdot \sqrt[4]{\prod_{n=1}^{4} p_n} \\ BP &= min(1, e^{1-r/c}), \end{aligned}$$

where p_n is the n-gram precision, c is the cardinality of the set of candidate sentences and r is the size of the smallest set of reference sentences.

To augment BLEU evaluation with paraphrasing information, we substitute each reference with the corresponding synthetic reference.

4.1.2 Data

We use the Chinese portion of the 2004 NIST MT dataset. This portion contains 200 Chinese documents, subdivided into a total of 1788 segments. Each segment is translated by ten machine translation systems and by four human translators. A quarter of the machine-translated segments are scored by human evaluators on a one-to-five scale along two dimensions: adequacy and fluency. We use only adequacy scores, which measure how well content is preserved in the translation.

4.1.3 Alternative Paraphrasing Techniques

To investigate the effect of paraphrase quality on automatic evaluation, we consider two alternative paraphrasing resources: Latent Semantic Analysis (LSA), and Brown clustering (Brown et al., 1992). These techniques are widely used in NLP applications, including language modeling, information extraction, and dialogue processing (Haghighi et al., 2005; Serafin and Eugenio, 2004; Miller et al., 2004). Both techniques are based on distributional similarity. The Brown clustering is computed by considering mutual information between adjacent words. LSA is a dimensionality reduction technique that projects a word co-occurrence matrix to lower dimensions. This lower dimensional representation is then used with standard similarity measures to cluster the data. Two words are considered to be a paraphrase pair if they appear in the same cluster.

We construct 1000 clusters employing the Brown method on 112 million words from the North American New York Times corpus. We keep the top 20 most frequent words for each cluster as paraphrases. To generate LSA paraphrases, we used the Infomap software[4] on a 34 million word collection of articles from the American News Text corpus. We used the default parameter settings: a 20,000 word vocabulary, the 1000 most frequent words (minus a stoplist) for features, a 15 word context window on either side of a word, a 100 feature reduced representation, and the 20 most similar words as paraphrases.

While we experimented with several parameter settings for LSA and Brown methods, we do not claim that the selected settings are necessarily optimal. However, these methods present sensible comparison points for understanding the relationship between paraphrase quality and its impact on automatic evaluation.

Table 3 shows synthetic references produced by the different paraphrasing methods.

[4] http://infomap-nlp.sourceforge.net

Method	1 reference	2 references
BLEU	0.9657	0.9743
WordNet	0.9674	0.9763
ContextWN	**0.9677**	**0.9764**
LSA	0.9652	0.9736
Brown	0.9662	0.9744

Table 4: Pearson adequacy correlation scores for rewriting using one and two references, averaged over ten runs.

Method	vs. BLEU	vs. ContextWN
WordNet	◁◁	△△
ContextWN	◁◁	-
LSA	X	△△
Brown	◁◁	△

Table 5: Paired t-test significance for all methods compared to BLEU as well as our method for one reference. Two triangles indicates significant at the 99% confidence level, one triangle at the 95% confidence level and X not significant. Triangles point towards the better method.

4.2 Impact of Paraphrases on Machine Translation Evaluation

The standard way to analyze the performance of an evaluation metric in machine translation is to compute the Pearson correlation between the automatic metric and human scores (Papineni et al., 2002; Koehn, 2004; Lin and Och, 2004; Stent et al., 2005). Pearson correlation estimates how linearly dependent two sets of values are. The Pearson correlation values range from 1, when the scores are perfectly linearly correlated, to -1, in the case of inversely correlated scores.

To calculate the Pearson correlation, we create a document by concatenating 300 segments. This strategy is commonly used in MT evaluation, because of BLEU's well-known problems with documents of small size (Papineni et al., 2002; Koehn, 2004). For each of the ten MT system translations,

Reference:	The monthly magazine "Choices" has won the deep trust of the residents. The current Internet edition of "Choices" will give full play to its functions and will help consumers get quick access to market information.
System:	The public has a lot of faith in the "Choice" monthly magazine and the Council is now working on a web version. This will enhance the magazine's function and help consumer to acquire more up-to-date market information.
WordNet	The monthly magazine "Choices" has won the deep **faith** of the residents. The current Internet **version** of "Choices" will give full play to its functions and will help consumers **acquire** quick access to market information.
ContextWN	The monthly magazine "Choices" has won the deep trust of the residents. The current Internet **version** of "Choices" will give full play to its functions and will help consumers **acquire** quick access to market information.
LSA	The monthly magazine "**Choice**" has won the deep trust of the residents. The current **web** edition of "**Choice**" will give full play to its functions and will help **consumer** get quick access to market information.
Brown	The monthly magazine "Choices" has won the deep trust of the residents. The current Internet **version** of "Choices" will give full play to its functions and will help consumers get quick access to market information.

Table 3: Sample of paraphrasings produced by each method based on the corresponding system translation. Paraphrased words are in bold and filtered words underlined.

the evaluation metric score is calculated on the document and the corresponding human adequacy score is calculated as the average human score over the segments. The Pearson correlation is calculated over these ten pairs (Papineni et al., 2002; Stent et al., 2005). This process is repeated for ten different documents created by the same process. Finally, a paired t-test is calculated over these ten different correlation scores to compute statistical significance.

Table 4 shows Pearson correlation scores for BLEU and the four paraphrased augmentations, averaged over ten runs.[5] In all ten tests, our method based on contextual rewriting (ContextWN) improves the correlation with human scores over BLEU. Moreover, in nine out of ten tests ContextWN outperforms the method based on WordNet. The results of statistical significance testing are summarized in Table 5. All the paraphrasing methods except LSA, exhibit higher correlation with human scores than plain BLEU. Our method significantly outperforms BLEU, and all the other paraphrase-based metrics. This consistent improvement confirms the importance of contextual filtering.

[5]Depending on the experimental setup, correlation values can vary widely. Our scores fall within the range of previous researchers (Papineni et al., 2002; Lin and Och, 2004).

The third column in Table 4 shows that automatic paraphrasing continues to improve correlation scores even when two human references are paraphrased using our method.

4.3 Evaluation of Paraphrase Quality

In the last section, we saw significant variations in MT evaluation performance when different paraphrasing methods were used to generate a synthetic reference. In this section, we examine the correlation between the quality of automatically generated paraphrases and their contribution to automatic evaluation. We analyze how the substitution frequency and the accuracy of those substitutions contributes to a method's performance.

We compute the substitution frequency of an automatic paraphrasing method by counting the number of words it rewrites in a set of reference sentences. Table 6 shows the substitution frequency and the corresponding BLEU score. The substitution frequency varies greatly across different methods — LSA is by far the most prolific rewriter, while Brown produces very few substitutions. As expected, the more paraphrases identified, the higher the BLEU score for the method. However, this increase does

Method	Score	Substitutions
BLEU	0.0913	-
WordNet	0.0969	994
ContextWN	0.0962	742
LSA	0.992	2080
Brown	0.921	117

Table 6: Scores and the number of substitutions made for all 1788 segments, averaged over the different MT system translations

Method	Judge 1 accuracy	Judge 2 accuracy	Kappa
WordNet	63.5%	62.5%	0.74
ContextWN	**75%**	**76.0%**	0.69
LSA	30%	31.5%	0.73
Brown	56%	56%	0.72

Table 7: Accuracy scores by two human judges as well as the Kappa coefficient of agreement.

not translate into better evaluation performance. For instance, our contextual filtering method removes approximately a quarter of the paraphrases suggested by WordNet and yields a better evaluation measure. These results suggest that the substitution frequency cannot predict the utility value of the paraphrasing method.

Accuracy measures the correctness of the proposed substitutions in the context of a reference sentence. To evaluate the accuracy of different paraphrasing methods, we randomly extracted 200 paraphrasing examples from each method. A paraphrase example consists of a reference sentence, a reference word to be paraphrased and a proposed paraphrase of that reference (that actually occurred in a corresponding system translation). The judge was instructed to mark a substitution as correct only if the substitution was both semantically and grammatically correct in the context of the original reference sentence.

Paraphrases produced by the four methods were judged by two native English speakers. The pairs were presented in random order, and the judges were not told which system produced a given pair. We employ a commonly used measure, Kappa, to assess agreement between the judges. We found that

	negative	positive
filtered	40	27
non-filtered	33	100

Table 8: Confusion matrix for the context filtering method on a random sample of 200 examples labeled by the first judge.

on all the four sets the Kappa value was around 0.7, which corresponds to substantial agreement (Landis and Koch, 1977).

As Table 7 shows, the ranking between the accuracy of the different paraphrasing methods mirrors the ranking of the corresponding MT evaluation methods shown in Table 4. The paraphrasing method with the highest accuracy, ContextWN, contributes most significantly to the evaluation performance of BLEU. Interestingly, even methods with moderate accuracy, i.e. 63% for WordNet, have a positive influence on the BLEU metric. At the same time, poor paraphrasing accuracy, such as LSA with 30%, does hurt the performance of automatic evaluation.

To further understand the contribution of contextual filtering, we compare the substitutions made by WordNet and ContextWN on the same set of sentences. Among the 200 paraphrases proposed by WordNet, 73 (36.5%) were identified as incorrect by human judges. As the confusion matrix in Table 8 shows, 40 (54.5%) were eliminated during the filtering step. At the same time, the filtering erroneously eliminates 27 positive examples (21%). Even at this level of false negatives, the filtering has an overall positive effect.

5 Conclusion and Future Work

This paper presents a comprehensive study of the impact of paraphrases on the accuracy of automatic evaluation. We found a strong connection between the quality of automatic paraphrases as judged by humans and their contribution to automatic evaluation. These results have two important implications: (1) refining standard measures such as BLEU with paraphrase information moves the automatic evaluation closer to human evaluation and (2) applying paraphrases to MT evaluation provides a task-based assessment for paraphrasing accuracy.

We also introduce a novel paraphrasing method based on contextual substitution. By posing the paraphrasing problem as a discriminative task, we can incorporate a wide range of features that improve the paraphrasing accuracy. Our experiments show improvement of the accuracy of WordNet paraphrasing and we believe that this method can similarly benefit other approaches that use lexico-semantic resources to obtain paraphrases.

Our ultimate goal is to develop a contextual filtering method that does not require candidate selection based on a lexico-semantic resource. One source of possible improvement lies in exploring more powerful learning frameworks and more sophisticated linguistic representations. Incorporating syntactic dependencies and class-based features into the context representation could also increase the accuracy and the coverage of the method. Our current method only implements rewriting at the word level. In the future, we would like to incorporate substitutions at the level of phrases and syntactic trees.

Acknowledgments

The authors acknowledge the support of the National Science Foundation (Barzilay; CAREER grant IIS-0448168) and DARPA (Kauchak; grant HR0011-06-C-0023). Thanks to Michael Collins, Charles Elkan, Yoong Keok Lee, Philip Koehn, Igor Malioutov, Ben Snyder and the anonymous reviewers for helpful comments and suggestions. Any opinions, findings and conclusions expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or NSF.

References

B. Babych, A. Hartley. 2004. Extending the BLEU evaluation method with frequency weightings. In *Proceedings of the ACL*, 621–628.

S. Banerjee, A. Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*, 65–72.

R. Barzilay, L. Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of NAACL-HLT*, 16–23.

P. F. Brown, P. V. deSouza, R. L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479.

I. Dagan, O. Glickman, B. Magnini, eds. 2005. *The PASCAL recognizing textual entailment challenge*, 2005.

P. Edmonds, G. Hirst. 2002. Near synonymy and lexical choice. *Computational Linguistics*, 28(2):105–144.

A. Haghighi, A. Ng, C. Manning. 2005. Robust textual inference via graph matching. In *Proceedings of NAACL-HLT*, 387–394.

S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Girju, V. Rus, P. Morarescu. 2001. The role of lexico-semantic feedback in open-domain textual question-answering. In *Proceedings of ACL*, 274–291.

P. Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*, 388–395.

J. R. Landis, G. G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33:159–174.

C. Lin, F. Och. 2004. ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of COLING*, 501–507.

I. D. Melamed, R. Green, J. P. Turian. 2003. Precision and recall of machine translation. In *Proceedings of NAACL-HLT*, 61–63.

S. Miller, J. Guinness, A. Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proceedings of HLT-NAACL*, 337–342.

NIST. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics, 2002.

B. Pang, K. Knight, D. Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of NAACL-HLT*, 102–209.

K. Papineni, S. Roukos, T. Ward, W. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the ACL*, 311–318.

C. Quirk, C. Brockett, W. Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proceedings of EMNLP*, 142–149.

R. E. Schapire, Y. Singer. 2000. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168.

R. Serafin, B. D. Eugenio. 2004. FLSA: Extending latent semantic analysis with features for dialogue act classification. In *Proceedings of the ACL*, 692–699.

A. Stent, M. Marge, M. Singhai. 2005. Evaluating evaluation methods for generation in the presense of variation. In *Proceedings of CICLING*, 341–351.

An Information-Theoretic Approach to Automatic Evaluation of Summaries

Chin-Yew Lin[+], Guihong Cao[*], Jianfeng Gao[#], and Jian-Yun Nie[*]

Information Sciences Institute[+]
University of Southern California
Marina del Rey, CA 90292
USA
cyl@isi.edu

Université de Montréal[*]
Montréal, Canada
{caogui,nie}@iro.umontreal.ca

Microsoft Corporation[#]
One Microsoft Way
Redmond, WA 98052
USA
jfgao@microsoft.com

Abstract

Until recently there are no common, convenient, and repeatable evaluation methods that could be easily applied to support fast turn-around development of automatic text summarization systems. In this paper, we introduce an information-theoretic approach to automatic evaluation of summaries based on the Jensen-Shannon divergence of distributions between an automatic summary and a set of reference summaries. Several variants of the approach are also considered and compared. The results indicate that JS divergence-based evaluation method achieves comparable performance with the common automatic evaluation method ROUGE in single documents summarization task; while achieves better performance than ROUGE in multiple document summarization task.

1 Introduction

Most previous automatic evaluation methods in summarization use co-occurrence statistics (Lin and Hovy 2003) to measure the content overlap between an automatic summary and a set of reference summaries. Among them, ROUGE (Lin 2004) has been used by the annual summarization evaluation conference, Document Understanding Conference[1] (DUC), sponsored by NIST since 2001. Content and quality of a summary are the two main aspects of summarization measured in the past DUCs. Using a manual evaluation interface called SEE[2], NIST assessors compared the content overlap between a system summary and a reference summary and assigned a coverage score to indicate the extent of the overlap between system and reference summaries. The overall system content coverage score was then the average of coverage scores over a set of test topics. NIST assessors also judged the quality of a peer summary by answering a set of quality assessment questions related to grammaticality, coherence, and organization for each system summary. However, we only focus on automatic evaluation of content coverage in this paper and aim at establishing a statistical framework that can perform at least as good as the current state-of-the-art automatic summarization evaluation methods such as ROUGE.

We start with a brief description of our statistical summary generation model and how to estimate its parameters in the next section. We then describe experimental setups and criterion of success in Section 3. The results of the experiments are shown and analyzed in Section 4. We discuss related work and recent advances in statistical language models for information retrieval in Section 5. Finally, we conclude and suggest future directions in Section 6.

2 Summarization Evaluation Using Information-Theoretic Measures

Given a set of documents $D = \{d_1, d_2, \ldots, d_i\}$[3], $i = 1$ to n, we assume there exists a probabilistic distribution with parameters specified by θ_R that generates reference summaries from D. The task of summarization is to estimate θ_R. Similarly, we as-

[1] Please see: http://duc.nist.gov for more information.

[2] SEE can be downloaded at: http://www.isi.edu/~cyl/SEE.

[3] $n = 1$ for a single document summarization task; $n > 1$ for a multi-document summarization task.

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 463–470,
New York, June 2006.

sume every system summary is generated from a probabilistic distribution with parameters specified by θ_A. Therefore, a good summarizer should have its θ_A very close to θ_R and the process of summary evaluation could be viewed as a task of estimating the distance between θ_A and θ_R.

For example, if we use Kullback-Leibler (*KL*) divergence as the distance function, then the summary evaluation task could be viewed as finding the *KL* divergence between θ_A and θ_R. However, *KL* divergence is unspecified when a value is defined in θ_R but not in θ_A (Lin 1991, Dagan et al. 1999). Usually smoothing has to be applied to address this missing data problem (unseen word in this case).

Another problem is that *KL* divergence is not symmetric, i.e. $KL(\theta_R \| \theta_A) \neq KL((\theta_A \| \theta_R)$, except when $\theta_R = \theta_A$. This is counter-intuitive in our application scenario. We therefore use generalized Jensen-Shannon (*JS*) divergence proposed by Lin (1991). The *JS* divergence can be written as follows:

$$JS_\pi(p_1, p_2, \cdots, p_n) = H\left(\sum_{i=1}^{n} \pi_i p_i\right) - \sum_{i=1}^{n} \pi_i H(p_i), \quad (1)$$

where p_i is a probability distribution with weight π_i, $\sum_{i=1}^{n} \pi_i = 1$, and $H(\cdot)$ is Shannon entropy. For $n = 2$ and equal weight, we have the following:

$$JS_{1/2}(p_1 \| p_2) = \frac{1}{2} \sum_{\theta_i} \left(p_1 \log\left(\frac{p_1}{\frac{1}{2}p_1 + \frac{1}{2}p_2} \right) + p_2 \log\left(\frac{p_2}{\frac{1}{2}p_1 + \frac{1}{2}p_2} \right) \right) \quad (2)$$

Since the Jensen-Shannon divergence is a distance measure, we take its negative value to indicate the similarity between two distributions as follows:

$$Score^{JS}_{summary}(S_A \mid S_R) = -JS_{1/2}(p(\theta_A \mid S_A) \| p(\theta_R \mid S_R)). \quad (3)$$

Equation (3) suggests that the problem of summary evaluation could be cast as ranking system summaries according to their negative Jensen-Shannon divergence with respect to the estimated posterior distribution of reference summaries. The question now is how to estimate these distributions.

2.1 Estimation of Posterior and Prior System Summary Distributions

θ_A is estimated via maximum *a posterior* (MAP) as:

$$\theta_A^{MP} = \arg\max_{\theta_A} p(\theta_A \mid S_A)$$

By Bayes' rule, the posterior probability of θ_A given S_A, $p(\theta_A \mid S_A)$, can be written as:

$$p(\theta_A \mid S_A) = \frac{p(S_A \mid \theta_A) p(\theta_A)}{p(S_A)}, \quad (4)$$

Assuming a multinomial generation model (Zaragoza et al. 2003) for each summary, parameterized by:

$$\theta_A = (\theta_{A,1}, \theta_{A,2}, ..., \theta_{A,m}) \in [0,1]^m, \quad \sum_{i=1}^{m} \theta_{A,i} = 1.$$

$\theta_{A,i}$ is the parameter of generating word *i* in summary S_A and *m* is the total number of words in the vocabulary. Assuming a bag-of-words unigram model, the system summary likelihood can be expressed as follows:

$$p(S_A \mid \theta_A) = Z_{a_0} \prod_{i=1}^{m} (\theta_{A,i})^{a_i}, \quad (5)$$

where a_i is the number of word *i* occurring in summary S_A, $a_0 = \sum_{i=1}^{m} a_i$, and Z_{a_0} is a constant as:

$$Z_{a_0} = \frac{\Gamma(a_0 + 1)}{\prod_{i=1}^{m} \Gamma(a_i + 1)},$$

where Γ is the gamma function, i.e. $\Gamma(n+1) = n\Gamma(n)$, $\Gamma(0) = 1$, *n* is an integer and $n \geq 0$.

In a MAP estimate, we usually choose the *conjugate* distribution of the generation distribution for a prior. In our case, we assume a Dirichlet prior distribution (the conjugate distribution of the multinomial distribution) as follows:

$$p(\theta_A) = Z'_{\alpha_0} \prod_{i=1}^{m} (\theta_{A,i})^{\alpha_i - 1}, \quad (6)$$

where α_i is hyperparameter related to word *i*, $\alpha_0 = \sum_{i=1}^{m} \alpha_i$, $\alpha_i > 0$, and Z'_{α_0} is:

$$Z'_{\alpha_0} = \frac{\Gamma(\alpha_0)}{\prod_{i=1}^{m} \Gamma(\alpha_i)}.$$

By the theory of total probability, the system summary probability can be computed as follows:

$$\begin{aligned} p(S_A) &= \int_\Theta p(S_A \mid \theta_A) p(\theta_A) d\theta_A \\ &= Z_{a_0} Z'_{\alpha_0} \int_\Theta \prod_{i=1}^{m} (\theta_{A,i})^{a_i + \alpha_i - 1} d\theta_A \\ &= Z_{a_0} Z'_{\alpha_0} \frac{\prod_{i=1}^{m} \Gamma(a_i + \alpha_i)}{\Gamma(a_0 + \alpha_0)}. \end{aligned} \quad (7)$$

Substituting (5), (6), and (7) into (4), we have the posterior distribution $p(\theta_A \mid S_A)$ as below:

$$\begin{aligned} & p(\theta_A \mid S_A) \\ &= \frac{p(S_A \mid \theta_A)p(\theta_A)}{p(S_A)} \\ &= \frac{\Gamma(a_0+\alpha_0)}{\prod_{i=1}^{m}\Gamma(a_i+\alpha_i)}\prod_{i=1}^{m}(\theta_{A,i})^{a_i+\alpha_i-1} \\ &= Z'_{a_0+\alpha_0}\prod_{i=1}^{m}(\theta_{A,i})^{a_i+\alpha_i-1}. \end{aligned} \quad (8)$$

We now turn to discuss different ways of estimating $\theta_{A,i}$ and α_i and their implications as described by Zaragoza et al. (2003).

According to Equation (8), the posterior distribution of θ_A given S_A is also a Dirichlet distribution. Its maximum posterior estimation has the following form (Gelman et al. 2003):

$$\theta_{A,i}^{MP} = \frac{a_i+\alpha_i-1}{a_0+\alpha_0-m}, \quad (9)$$

and the posterior distribution (8) can be written as:

$$p(\theta_A \mid S_A) \approx p(\theta_A^{MP} \mid S_A) = Z'_{a_0+\alpha_0}\prod_{i=1}^{m}(\theta_{A,i}^{MP})^{a_i+\alpha_i-1}. \quad (10)$$

If we set $\alpha_i = 1$, then $\theta_{A,i}$ does not depend on α_i, i.e. all possible θ_A's have equal prior. In this case, equation (9) becomes the maximum likelihood estimation as follows:

$$\theta_{A,i}^{ML} = \frac{a_i}{a_0} \quad (11)$$

and the posterior distribution (8) can be written as:

$$p(\theta_A \mid S_A) \approx p(\theta_A^{ML} \mid S_A) = Z'_{a_0+m}\prod_{i=1}^{m}(\theta_{A,i}^{ML})^{a_i}. \quad (12)$$

The problem with using maximum likelihood estimation is when a_i equal to zero. If zero occurrence happens for word *i*, then its maximum likelihood estimation, $\theta_{A,i}^{ML}$, would be zero and the whole posterior distribution would be zero. To tackle this problem, we need to redistribute some probability mass to zero occurrence events or unseen word events. The process of redistribution is called *smoothing* in the language modeling literatures. For an in-depth discussion of this topic, please see Chen and Goodman (1996).

By choosing different value for α_i, we could derive different smoothing methods as discussed in Zaragoza et al. (2003). For example, we could estimate α_i using topic collection frequency by setting $\alpha_i = \mu p(w_i \mid T)+1$, where μ is a scaling factor and $p(w_i \mid T)$ is the probability of word *i* occurring in topic *T*. This is called Bayes-smoothing and has been used in language modeling for information retrieval (Zhai and Lafferty 2004). The Bayes-smoothing can be written as:

$$\theta_{A,i}^{BS} = \frac{a_i+\mu p(w_i \mid T)}{a_0+\mu} \quad (13)$$

Using Equation (13), Equation (8) becomes:

$$p(\theta_A \mid S_A) \approx p(\theta_A^{BS} \mid S_A) = Z'_{a_i+\mu+m}\prod_{i=1}^{m}(\theta_{A,i}^{BS})^{a_i+\mu p(w_i \mid T)}. \quad (14)$$

We now turn to estimating the posterior distribution of θ_R given a reference summary S_R.

2.2 Estimation of Reference Summary Distributions

Given a reference summary S_R, we could estimate posterior distribution θ_R in the same way that we estimate posterior distribution θ_A as follows:

$$\begin{aligned} p(\theta_R \mid S_R) &= \frac{p(S_R \mid \theta_R)p(\theta_R)}{p(S_R)} \\ &= Z'_{a_0+\alpha_0}\prod_{i=1}^{m}(\theta_{R,i})^{a_i+\alpha_i-1}, \end{aligned} \quad (15)$$

where a_i is the number of occurrence of word *i* in reference summary S_R.

Given another reference summary S'_R, i.e., when multiple reference summaries are available, the posterior distribution can be updated using Bayesian inference as follows:

$$\begin{aligned} & p(\theta_R \mid S_R, S'_R) \\ &= \frac{p(\theta_R, S'_R \mid S_R)}{p(S'_R \mid S_R)} \\ &= \frac{p(S'_R \mid \theta_R, S_R)p(\theta_R \mid S_R)}{p(S'_R \mid S_R)} \\ &= \frac{p(S'_R \mid \theta_R)p(\theta_R \mid S_R)}{p(S'_R)} \\ &= \frac{\left(Z_{a'_0}\prod_{i=1}^{m}(\theta_{R,i})^{a'_i}\right)\left(Z'_{a_0+\alpha_0}\prod_{i=1}^{m}(\theta_{R,i})^{a_i+\alpha_i-1}\right)}{Z_{a'_0}Z'_{a_0+\alpha_0}\dfrac{\prod_{i=1}^{m}\Gamma(a'_i+a_i+\alpha_i)}{\Gamma(a'_0+a_0+\alpha_0)}} \\ &= \frac{\Gamma(a'_0+a_0+\alpha_0)}{\prod_{i=1}^{m}\Gamma(a'_i+a_i+\alpha_i)}\prod_{i=1}^{m}(\theta_{T,i})^{a'_i+a_i+\alpha_i-1} \\ &= Z'_{a'_0+a_0+\alpha_0}\prod_{i=1}^{m}(\theta_{R,i})^{a'_i+a_i+\alpha_i-1}, \end{aligned} \quad (16)$$

where $p(\theta_R \mid S_R)$ is the posterior distribution from equation (15), S'_R is independent of S_R, and $p(S'_R)$ is computed using equation (7) but with the posterior distribution $p(\theta_R \mid S_R)$ as prior. In a more general case, given multiple (*L*) reference summaries, $S_{R,1}, \cdots, S_{R,L}$, the posterior distribution of θ_R could

be written as follows by repeat application of Bayesian inference with equation (16):

$$p(\theta_R \mid S_{R,1}, \cdots, S_{R,L}) = Z'_{\alpha_0 + \sum_{j=1}^{L} a_{0,j}} \prod_{i=1}^{m} (\theta_{R,i})^{\alpha_i - 1 + \sum_{j=1}^{L} a_{i,j}}, \quad (17)$$

where $a_{i,j}$ is the number of occurrence of word *i* in reference summary $S_{R,j}$, and

$$a_{0,j} = \sum_{i=0}^{m} a_{i,j}. \quad (18)$$

Equation (18) is the total number of words in reference summary $S_{R,j}$. The total number of words in the topic collection could be computed as follows:

$$\sum_{j=1}^{L} a_{0,j} = \sum_{j=1}^{L} \sum_{i=1}^{m} a_{i,j}. \quad (19)$$

Equation (17) indicates that estimation of posterior distribution given multiple summaries is the same as estimation of posterior distribution given a single summary that contains all the reference summaries. This is reasonable since we assume a bag-of-word unigram likelihood model for generating summaries[4]. It also bodes well with the consensus-oriented manual summarization evaluation approaches proposed by van Halteren and Teufel (2003) and Nenkova and Passonneau (2004). With equations (8) and (17), the summary score of system summary, S_A, can be computed using Jensen-Shannon divergence from equation (3) as follows:

$$Score^{JS}_{summary}(S_A \mid S_R^{1,L}) = -JS_{1/2}(p(\theta_A \mid S_A) \,\square\, p(\theta_R \mid S_R^{1,L})), \quad (20)$$

where $S_R^{1,L}$ is a shorthand for $S_{R,1}, \cdots, S_{R,L}$.

3 Experimental Setup

We used data from DUC 2002 100-word single and multi-document tasks as our testing corpus. DUC 2002 data includes 59 topic sets. Each topic set contains about 10 news article pertaining to some news topic, for example, topic D061 is about "*Hurricane Gilbert*". Two human written summaries per topic are provided as reference summaries. 14 sets of system summaries and 1 simple lead baseline summary are included for the single document summarization task (total 15 runs); while 8 sets of system summaries, 1 lead baseline, and 1 latest news baseline are included for the multi-document summarization task (total 12 runs). All summaries are about 100 words[5]. Manually evaluation results in average coverage[6] scores are also included in the DUC 2002 data set.

The commonly used criterion of success to evaluate an automatic evaluation method is to compute the correlation between the ranking of systems according to human assigned scores and the ranking according to automatic scores (Papineni et al. 2002; Lin & Hovy 2003). We followed the same convention and computed Pearson's product moment correlation coefficient and Spearman's rank correlation coefficient as indicators of success.

Besides evaluating the performance of the automatic evaluation measure based on Jensen-Shannon (*JS*) divergence as defined in equation (2), we also compared it with measures based on *KL*-divergence and simple log likelihood. The effect of smoothing and the difference of using single and multiple reference summaries were also investigated. To examine the effect of using longer n-grams ($n > 1$), we also used bag-of-bigram and bag-of-trigram models by simply replace unigrams in the model proposed in Section 2 with bigrams and trigrams and treat them as unigrams. Lemur toolkit version 4.0[7] was used to estimate models with modification to speedup computation of bigram and trigram models. We also ran standard ROUGE v1.5.5 with ROUGE1 to 4 as baselines.

All experiments were run with common words excluded and Porter stemmer applied. We summarize these experiments in the following sections.

3.1 Jensen-Shannon Divergence (JSD)

We use equation (22) to compute summary score and apply maximum likelihood estimation (θ^{ML}) of the parameters according to equation (11). Using a unigram model and single reference summary, we rewrite equation (22) as follows:

[4] Please refer to equation (5).

[5] There were also 10-, 50-, and 200-word summary tasks in DUC 2002 multi-document summarization evaluation. However, we only used the data of 100-word summarization subtask for this experiment.

[6] Coverage is a weighted recall metric measuring the overlap of content between a system summary and a reference summary. For example, if 4 elementary discourse units (EDU, Marcu 1998) in a system summary partially match EDUs in a reference summary of total 8 EDUs, then its coverage score is R*4/8. R is the ratio of the partial match. R is 20%, 40%, 60%, 80%, or 100% in DUC 2002. A single human assigned the ratio using only one reference. The other reference was not used in the manual evaluation.

[7] The Lemur project: http://www.lemurproject.org.

$$Score^{JSD}_{summary}(S_A \mid S_R^{1,1}) =$$

$$-\frac{1}{2}\sum_{\theta_i^{MP}}\left(\begin{array}{l} p(\theta_i^{MP} \mid S_A)\log\left(\dfrac{p(\theta_i^{MP} \mid S_A)}{\frac{1}{2}p(\theta_i^{MP} \mid S_A)+\frac{1}{2}p(\theta_i^{MP} \mid S_R^{1,1})}\right)+ \\ p(\theta_i^{MP} \mid S_R^{1,1})\log\left(\dfrac{p(\theta_i^{MP} \mid S_R^{1,1})}{\frac{1}{2}p(\theta_i^{MP} \mid S_A)+\frac{1}{2}p(\theta_i^{MP} \mid S_R^{1,1})}\right)\end{array}\right)$$

where $p(\theta_i^{MP} \mid S_A)$ and $p(\theta_i^{MP} \mid S_R^{1,1})$ are estimated as follows:

$$\begin{aligned} p(\theta_i^{MP} \mid S_A) &= \frac{a_{A,i}}{a_{A,0}} \\ &= \frac{C(w_i,S_A)}{\sum_{w_i} C(w_i,S_A)}, \\ p(\theta_i^{MP} \mid S_R^{1,1}) &= \frac{a_{R,i}}{a_{R,0}} \\ &= \frac{C(w_i,S_R^{1,1})}{\sum_{w_i} C(w_i,S_R^{1,1})}. \end{aligned}$$

$C(w_i,S_A)$ and $C(w_i,S_R^{1,1})$ are the counts of word w_i in system summary S_A and reference summary $S_R^{1,1}$ respectively. When multiple reference summaries are used, $p(\theta_i^{MP} \mid S_R^{1,L})$ is estimated as follows:

$$\begin{aligned} p(\theta_i^{MP} \mid S_R^{1,L}) &= \frac{a_{R,i}^{1,L}}{a_{R,0}^{1,L}} \\ &= \frac{\sum_j C(w_i,S_{R,i}^{j,j})}{\sum_{w_i}\sum_j C(w_i,S_{R,i}^{j,j})}. \end{aligned}$$

3.2 Jensen-Shannon Divergence with Smoothing (JSDS)

To examine the effect of smoothing when we compute summary score using equation (22), we apply Bayes-smoothing as shown in equation (15). Using a unigram model and single reference summary, we rewrite equation (22) as follows:

$$Score^{JSDS}_{summary}(S_A \mid S_R^{1,1}) =$$

$$-\frac{1}{2}\sum_{\theta_i^{BS}}\left(\begin{array}{l} p(\theta_i^{BS} \mid S_A)\log\left(\dfrac{p(\theta_i^{BS} \mid S_A)}{\frac{1}{2}p(\theta_i^{BS} \mid S_A)+\frac{1}{2}p(\theta_i^{BS} \mid S_R^{1,1})}\right)+ \\ p(\theta_i^{BS} \mid S_R^{1,1})\log\left(\dfrac{p(\theta_i^{BS} \mid S_R^{1,1})}{\frac{1}{2}p(\theta_i^{BS} \mid S_A)+\frac{1}{2}p(\theta_i^{BS} \mid S_R^{1,1})}\right)\end{array}\right)$$

where $p(\theta_i^{BS} \mid S_A)$ and $p(\theta_i^{BS} \mid S_R^{1,1})$ are estimated as follows:

$$\begin{aligned} p(\theta_i^{BS} \mid S_A) &= \frac{a_{A,i}+\mu p(w_i \mid C)}{a_{A,0}+\mu} \\ &= \frac{C(w_i,S_A)+\mu p(w_i \mid C)}{\left(\sum_{w_i} C(w_i,S_A)\right)+\mu}, \\ p(\theta_i^{BS} \mid S_R^{1,1}) &= \frac{a_{R,i}+\mu p(w_i \mid C)}{a_{R,0}+\mu} \\ &= \frac{C(w_i,S_R^{1,1})+\mu p(w_i \mid C)}{\left(\sum_{w_i} C(w_i,S_R^{1,1})\right)+\mu}. \end{aligned}$$

$C(w_i,S_A)$ and $C(w_i,S_R^{1,1})$ are the counts of word w_i in system summary S_A and reference summary $S_R^{1,1}$ respectively. The Bayes-smoothing probability or Bayesian prior $p(w_i \mid C)$ is estimated from a general English corpus instead of the topic collection as we described in section 2.1. In our experiments, we used TREC AP88-90 collection that contained more than 200,000 news articles. When multiple reference summaries are used, $p(\theta_i^{BS} \mid S_R^{1,L})$ is estimated as follows:

$$\begin{aligned} p(\theta_i^{BS} \mid S_R^{1,L}) &= \frac{a_{R,i}^{1,L}+\mu p(w_i \mid C)}{a_{R,0}^{1,L}+\mu} \\ &= \frac{\left(\sum_j C(w_i,S_{R,i}^{j,j})\right)+\mu p(w_i \mid C)}{\left(\sum_{w_i}\sum_j C(w_i,S_{R,i}^{j,j})\right)+\mu}. \end{aligned}$$

The value of μ could be determined empirically. In this experiment we set μ to 2,000 following Zhai and Lafferty (2004).

3.3 Kullback-Leibler Divergence with Smoothing (KLDS)

To compare the performance of *JSD* and *JSDS* scoring methods with other alternative distance measure, we also compute summary scores using *KL* divergence with Bayes-smoothing as follows:

$$Score^{KL}_{summary}(S_A \mid S_R^{1,L}) = -\sum_{\theta_i^{BS}} p(\theta_i^{BS} \mid S_A)\log\left(\frac{p(\theta_i^{BS} \mid S_A)}{p(\theta_i^{BS} \mid S_R^{1,L})}\right)$$

The Bayes-smoothing factor μ is also set to 2,000 and θ_i^{BS} is estimated by the same way that we compute *JSDS*.

Unigram		JSD		JSDS		KLDS		LLS	
		P	S	P	S	P	S	P	S
SD	SR	**0.97**	**0.91**	0.61	0.25	0.59	0.23	-0.54	0.16
	MR	**0.97**	**0.91**	0.62	0.65	0.61	0.25	-0.60	0.11
MD	SR	**0.80**	**0.83**	0.44	0.64	0.34	0.54	0.21	0.36
	MR	**0.88**	**0.89**	0.76	0.81	0.61	0.71	0.47	0.60

Bigram		JSD		JSDS		KLDS		LLS	
		P	S	P	S	P	S	P	S
SD	SR	**0.92**	**0.90**	0.64	0.60	0.50	0.20	-0.81	0.05
	MR	**0.94**	**0.90**	0.64	0.62	0.53	0.26	-0.80	0.06
MD	SR	**0.91**	**0.88**	-0.17	-0.19	0.01	0.14	0.82	0.87
	MR	**0.96**	**0.94**	0.17	0.36	0.12	0.22	0.85	0.89

Trigram		JSD		JSDS		KLDS		LLS	
		P	S	P	S	P	S	P	S
SD	SR	**0.92**	**0.90**	0.68	0.44	0.53	0.11	-0.72	0.03
	MR	**0.94**	**0.90**	0.68	0.58	0.55	0.20	-0.71	0.01
MD	SR	**0.87**	**0.82**	-0.39	-0.33	-0.11	-0.10	0.50	0.54
	MR	**0.93**	**0.89**	-0.30	-0.26	-0.11	-0.10	0.54	0.54

Table 1. DUC 2002 single (SD) and multi-document summarization (MD) tasks' Pearson's (P) and Spearman's (S) correlations of automatic measures (*JSD*, *JSDS*, *KLDS*, and *LLS*) using single (SR) or multiple (MR) reference summaries. (Unigram: bag-of-unigram model, Bigram: bag-of-bigram model, and Trigram: bag-of-trigram model)

3.4 Log Likelihood with Smoothing (LLS)

As a baseline measure, we also compute the log likelihood score of an automatic summary given a reference summary or a set of reference summaries as follows:

$$Score^{LL}_{summary}(S_A \mid S_R^{1,L}) = \sum_{i=1}^{|S_A|} \log p(\theta_i^{BS} \mid S_R^{1,L}),$$

where $|S_A|$ is the length of S_A and $p(\theta_i^{BS} \mid S_R^{1,L})$ is estimated as before.

4 Results

Table 1 shows the results of all runs. According to Table 1, automatic evaluation measure based on Jensen-Shannon Divergence without Bayes-smoothing (*JSD*) performed the best among all measures. Among them, *JSD* over the bag-of-unigram model achieved the best results in the single document summarization task (P-SD-MR: 0.97, S-SD-MR: 0.91); while the bag-of-bigram model achieved the best results in the multiple document summarization task (P-MD-MR: 0.96, S-MD-MR: 0.94). Although the bag-of-bigram model did not perform as well as the bag-of-unigram model in the single document summarization task, its Pearson (SD-MR: 0.94) and Spearman (SD-MR: 0.90) correlation values were still over 90% regardless of single or multiple references were used.

ROUGE		ROUGE-1		ROUGE-2		ROUGE-3		ROUGE-4	
		P	S	P	S	P	S	P	S
MR	SD	0.99	0.84	1.00	0.96	1.00	0.98	**1.00**	**0.99**
	MD	0.70	0.59	0.89	0.84	**0.92**	**0.85**	0.90	0.78

Table 2. DUC 2002 single (SD) and multi-document (MD) summarization tasks' Pearson's (P) and Spearman's (S) correlations of automatic measures (ROUGE1-4) using multiple (MR) reference summaries.

We also observed that using multiple references outperformed using only single reference. This is reasonable since we expect to estimate models better when more reference summaries are available.

Smoothed measures did not perform well. This is not a surprise due to the nature of summarization evaluation. Intuitively, only information presented in system and reference summaries should be considered for evaluation.

The *JSD*-based measure was also compared favorably to ROUGE in the multiple document summarization task as shown in Table 2. In particular, the *JSD*-based measure over bag-of-bigram model using multiple references achieved much better results in both Pearson's and Spearman's correlations than all versions of ROUGE. For single document summarization task, the *JSD*-based measure still achieved high correlations (90%+) though it was not as high as ROUGE2, 3, and 4.

5 Related Work

The approach described in this paper is most similar to the Bayesian extension in information retrieval (IR) work by Zaragoza et al. (2003). In their work, query likelihood model was presented as Bayesian inference. Other earlier language modeling (Rosenfeld 2002) work in information retrieval, especially the idea of modeling a document using bag-of-word unigram model, also inspire this work (Berger and Lafferty 1999, Lafferty and Zhai 2001

Statistical language models such as document language model (Ponte and Croft 1998, Zhai and Lafferty 2004), relevance-based language models (Lavrenko and Croft 2001), and dependency-based language models (Gao et al. 2004) have been applied successfully in information retrieval. It has also been applied to topic detection and tracking (Lavrenko et al. 2002, Larkey et al. 2004). Ex-

tended models also have been developed to deal with vocabulary mismatch and query expansion problems (Berger and Lafferty 1999, Hofmann 1999, Lafferty and Zhai 2001). However, it has not been applied in automatic evaluation of summarization. Hori et al. (2004) also considered using "posterior probability" derived from consensus among human summaries as weighting factor to improve evaluations of speech summarization. But their notion of "posterior probability" was not true probability and was not presented as an integral part of the Bayesian inference framework as we have described in this paper.

6 Conclusions and Future Work

The research proposed in this paper aims at providing a pilot study of applying information-theoretic measures in automatic evaluation of summaries. With the initial success of this study, we would like to: (1) verify the results with other set of data, for example, DUC 2003 data, (2) tune the Bayesian smoothing parameter μ to further examine the effect of smoothing, (3) develop better content generation model and (4) add synonym and paraphrase matching capability in the future. To address (3), for example, we would like to explore mutual information-based dependency language modeling as proposed by Gao et al. (2004).

For (4), manual evaluation methods recently proposed separately by van Halteren and Teufel (2003), the factoid method, and Nenkova and Passouneau (2004), the pyramid method, tried to take advantage of the availability of multiple references. Both methods assume that the more important a piece of information is, the more reference summaries it appears in. These manual evaluation methods can identify semantic equivalents. For example, a summary content unit (SCU) "The diamonds were replaced by fake replicas[8]" created as defined in Nenkova and Passouneau (2004) from the following four contributing clauses (1a – d):

1. Authorities, responding to a tip, [*switched the diamonds with fakes*]$_{1a}$ and were waiting inside the building dressed as cleaners when the thieves burst in with a bulldozer and sledge-hammers.
2. However, authorities were tipped off and [*switched the diamonds with fakes*] $_{1b}$.
3. They disguised themselves as cleaners at the Millennium Dome, [*switched the diamonds with worthless glass*] $_{1c}$, and waited for the robbers, who planned to get away in a speedboat down the Thames River.
4. [*The diamonds had been swapped with glass replicas*] $_{1d}$.

Contributors (1a – d) from 4 reference summaries to the SCU are underlined. The manual pyramid method can identify these semantic equivalents. It is obvious that automatic evaluation methods relying on strict n-gram or lexical matching would only find two out of four possible matches, i.e. "*switched the diamonds with fakes*" from (1a) and (1b) while leave "*switched the diamonds with worthless glass*" (1c) and "*The diamonds had been swapped with glass replicas*" (1d) unmatched. Allowing near synonyms such as *fakes*, *worthless glass*, and *glass replicas* to match might help, but how to acquire these equivalents and how to assign appropriate weights to reflect their subtle differences remain open questions. To find semantic equivalents automatically, we would like to try query expansion techniques (Hofmann 1999, Lafferty and Zhai 2001, Bai et al. 2005, Cao et al. 2005) commonly used in IR. Proper query expansion boosts IR system performance. We suspect that these techniques would help a little but we probably would need to develop much better paraphrase expansion and matching techniques to see significant boost in overall performance.

7 Acknowledgement

Part of this work was conducted while the first author visited Microsoft Research Asia (MSRA) in the summer of 2005. He would like to thank Ming Zhou, Hsiao-Wuen Hon, and other staffs at MSRA for providing an excellent research environment and exciting intellectual exchanges during his visit.

Reference

Bai, Jing, Dawei Song, Peter Bruza, Jian-Yun Nie, and Guihong Cao. 2005. Query Expansion Using Term Relationships in Language Models for Information Retrieval. *Proceedings of International Conference on Information and Knowledge Management (CIKM) 2005*. October 31 – November 5, 2005, Bremen, Germany.

[8] Example is taken from Multilingual Summarization Evaluation 2005 (MSE2005), topic number 33003.

Berger, Adam and John Lafftery. 1999. *Information Retrieval as Statistical Translation*. Proceedings of ACM-SIGIR 1999. August 15-19, 1999, Berkeley, CA, USA.

Cao, Guihong, Jian-Yun Nie, and Jing Bai. 2005. Integrating Word Relationships into Language Models. *Proceedings of SIGIR 2005*. August 15-19, 2005, Salvador, Brazil.

Chen, Stanley and Joshua Goodman. 1996. An Empirical Study of Smoothing Techniques for Language Modeling. *Proceedings of 34th Annual Meeting on Association for Computational Linguistics*, page 310-318, June 23-28, Santa Cruz, California, USA.

Dagan, Ido, Lillian Lee, and Fernado C. N. Pereira. 1999. Similarity-Based Models of Word Cooccurrence Probability. *Machine Learning*. Vol 34, page 43-69, 1999.

Gao, Jianfeng, Jian-Yun Nie, Guangyuan Wu, and Guihong Cao. 2004. Dependence Language Model for Information Retrieval. *Proceedings of SIGIR 2004*. July 25-29, 2004, Sheffield, UK.

Gelman, Andrew, John B. Carlin, Hal S. Stern, and Donald B. Rubin. 2003. *Bayesian Data Analysis*. 2nd Edition. Chapman & Hall/CRC.

Hofmann, Thomas. 1999. Probabilistic Latent Semantic Indexing. *Proceedings of ACM-SIGIR 1999*. August 15-19, 1999, Berkeley, CA, USA.

Hori, Chiori, Tsutomu Hirao, and Hideki Isozaki. 2004. Evaluation Measures Considering Sentence Concatenation for Automatic Summarization by Sentence or Word Extraction. *Proceedings of Workshop on Text Summarization Branches Out*. July 25, 2004, Barcelona, Spain.

Kraaij, Wessel, Martijn Spitters, and Martine van der Heijden. 2001. Combining a Mixture Language Model and Naïve Bayes for Multi-Document Summarisation. *Proceedings of DUC 2001*.

Lafferty, John and Chiangxiang Zhai. 2001. Document Language Models, Query Models, and Risk Minimization for Information Retrieval. *Proceedings of ACM-SIGIR 2001*. September 9-13, 2001, New Orleans, LA, USA.

Larkey, Leah S., Fangfang Feng, Margaret Connell, and Victor Larvrenko. 2004. Language-specific Models in Multilingual Topic Tracking. *Proceedings of ACM-SIGIR 2004*. July 25-29, 2004. Sheffield, UK.

Lavrenko, Victor, James Allan, Edward DeGuzman, Daniel LaFlamme, Veera Pollard, and Stephen Thomas. 2002. Relevance Models for Topic Detection and Tracking. *Proceedings of HLT 2002*. March 24-27, 2002, San Diego, CA, USA.

Lavrenko, Victor and W. Bruce Croft. 2001. Relevance-Based Language Models. *Proceedings of ACM-SIGIR 2001*. September 9-13, 2001, New Orleans, LA, USA.

Lin, Chin-Yew and Eduard Hovy. 2003. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. *Proceedings of HLT-NAACL-2003*. May 27-June 1, 2003, Edmonton, Canada.

Lin, Chin-Yew. 2004. ROUGE: a Package for Automatic Evaluation of Summaries. *Proceedings of Workshop on Text Summarization 2004*. July 21-26, 2004, Barcelona, Spain.

Lin, Jianhua. 1991. Divergence Measures Based on the Shannon Entropy. *IEEE Transactions on Information Theory*, 37(1), page 145-151, 1991.

Nenkova, Ani and Rebecca Passonneau. 2004. Evaluating Content Selection in Summarization: the Pyramid Method. *Proceedings of NAACL-HLT 2004*. May 2-7, 2004, Boston, MA, USA.

Papineni, Kishore, 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 (ACL-2002)*, Philadelphia, USA, July 2002, page 311-318.

Ponte, Jay M. and W. Bruce Croft. 1998. A language modeling approach to information retrieval. *Proceedings of ACM-SIGIR 1998*, pages 275-281. August 24-28, 1998, Melbourne, Australia.

Rosenfeld, Ronald. 2002. Two Decades of Statistical Language Modeling: Where do We Go from Here? *Proceedings of IEEE*.

Van Halteren, Hans and Simone Teufel. 2003. Examining the Consensus between Human Summaries: Initial Experiments with Factoid Analysis. *Proceedings of Workshop on Text Summarization 2003*. May 27-June 1, 2003, Edmonton, Canada.

Zaragoza, Hugo, Djoerd Hiemstra, and Michael Tipping. 2003. Bayesian Extension to the Language Model for Ad Hoc Information Retrieval. *Proceedings of ACM-SIGIR 2003*. July 28-August 1, 2003, Toronto, Canada.

Zhai, Chengxiang and John Lafferty. 2004. A Study of Smoothing Methods for Language Models Applied to Information Retrieval. *ACM Transactions on Information Systems*, Vol 22, No. 2, April 2004, pages 179-214.

Cross Linguistic Name Matching in English and Arabic: A "One to Many Mapping" Extension of the Levenshtein Edit Distance Algorithm

Dr. Andrew T. Freeman, Dr. Sherri L. Condon and Christopher M. Ackerman
The Mitre Corporation
7525 Colshire Dr
McLean, Va 22102-7505
{afreeman, scondon, cackerman}@mitre.org

Abstract

This paper presents a solution to the problem of matching personal names in English to the same names represented in Arabic script. Standard string comparison measures perform poorly on this task due to varying transliteration conventions in both languages and the fact that Arabic script does not usually represent short vowels. Significant improvement is achieved by augmenting the classic Levenshtein edit-distance algorithm with character equivalency classes.

1 Introduction to the problem

Personal names are problematic for all language technology that processes linguistic content, especially in applications such as information retrieval, document clustering, entity extraction, and translation. Name matching is not a trivial problem even within a language because names have more than one part, including titles, nicknames, and qualifiers such as *Jr*. or *II*. Across documents, instances of the name might not include the same name parts, and within documents, the second or third mention of a name will often have only one salient part. In multilingual applications, the problem is complicated by the fact that when a name is represented in a script different from its native script, there may be several alternative representations for each phoneme, leading to large number of potential variants for multi-part names.

A good example of the problem is the name of the current leader of Libya. In Arabic, there is only one way to write the consonants and long vowels of any person's name, and the current leader of Libya's name in un-vocalized Arabic text can only be written as معمر القذافي. In English, his name has many common representations. Table 1 documents the top five hits returned from a web search at www.google.com, using various English spellings of the name.

Version	Occurrences
Muammar Gaddafi	43,500
Muammar Qaddafi	35,900
Moammar Gadhafi	34,100
Muammar Qadhafi	15,000
Muammar al Qadhafi	11,500

Table 1. Qadhafy's names in English

Part of this variation is due to the lack of an English phoneme corresponding to the Standard Arabic phoneme /q/. The problem is further compounded by the fact that in many dialects spoken in the Arabic-speaking world, including Libya, this phoneme is pronounced as [g].

The engineering problem is how one reliably matches all versions of a particular name in language A to all possible versions of the same name in language B. Most solutions employ standard string similarity measures, which require the names to be represented in a common character set. The solution presented here exploits transliteration conventions in normalization procedures and equivalence mappings for the standard Levenshtein distance measure.

2 Fuzzy string matching

The term fuzzy matching is used to describe methods that match strings based on similarity rather than identity. Common fuzzy matching techniques include edit distance, n-gram matching, and normalization procedures such as Soundex.

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 471–478,
New York, June 2006.

This section surveys methods and tools currently used for fuzzy matching.

2.1 Soundex

Patented in 1918 by Odell and Russell the Soundex algorithm was designed to find spelling variations of names. Soundex represents classes of sounds that can be lumped together. The precise classes and algorithm are shown below in figures 1 and 2.

Code:	0	1	2	3	4	5	6
Letters:	aeiouy hw	bp fv	cgjkq sxz	dt	l	mn	r

Figure 1: Soundex phonetic codes

1. Replace all but the first letter of the string by its phonetic code.
2. Eliminate any adjacent repetitions of codes.
3. Eliminate all occurrences of code 0, i.e. eliminate all vowels.
4. Return the first four characters of the resulting string.
5. Examples: Patrick = P362, Peter = P36, Peterson = P3625

Figure 2: The Soundex algorithm

The examples in figure 2 demonstrate that many different names can appear to match each other when using the Soundex algorithm.

2.2 Levenshtein Edit Distance

The Levenshtein algorithm is a string edit-distance algorithm. A very comprehensive and accessible explanation of the Levenshtein algorithm is available on the web at http://www.merriampark.com/ld.htm.

The Levenshtein algorithm measures the edit distance where edit distance is defined as the number of insertions, deletions or substitutions required to make the two strings match. A score of zero represents a perfect match.

With two strings, string *s* of size *m* and string *t* of size *n*, the algorithm has O(*nm*) time and space complexity. A matrix is constructed with *n* rows and *m* columns. The function $e(s_i,t_j)$ where s_i is a character in the string *s,* and t_j is a character in string *t* returns a 0 if the two characters are equal and a 1 otherwise. The algorithm can be represented compactly with the recurrence relation shown in figure 3.

```
for each i from 0 to |s|
    for each j from 0 to |t|
        levenshtein(0; 0) = 0
        levenshtein(i; 0) = i
        levenshtein(0;j) = j
        levenshtein (i;j) =
                min[levenshtein (i − 1; j) + 1;
                levenshtein(i; j − 1) + 1;
                levenshtein(i − 1; j − 1) +
                e(s_i; t_j )]
```

Figure 3. Recurrence relation for Levenshtein edit distance

A simple "fuzzy-match" algorithm can be created by dividing the Levenshtein edit distance score by the length of the shortest (or longest) string, subtracting this number from one, and setting a threshold score that must be achieved in order for the strings to be considered a match. In this simple approach, longer pairs of strings are more likely to be matched than shorter pairs of strings with the same number of different characters.

2.3 Editex

The Editex algorithm is described by Zobel and Dart (1996). It combines a Soundex style algorithm with Levenshtein by replacing the $e(s_i,t_j)$ function of Levenshtein with a function $r(s_i,t_j)$. The function $r(s_i,t_j)$ returns 0 if the two letters are identical, 1 if they belong to the same letter group and 2 otherwise. The full algorithm with the letter groups is shown in figures 4 and 5. The Editex algorithm neutralizes the *h* and *w*. This shows up in the algorithm description as $d(s_{i-1},s_i)$. It is the same as $r(s_i,t_j)$, with two exceptions. It compares letters of the same string rather than letters from the different strings. The other difference is that if s_{i-1} is *h* or *w*, and $s_{i-1} \neq s_i$, then $d(s_{i-1},s_i)$ is one.

```
for each i from 0 to |s|
    for each j from 0 to |t|
    editex(0; 0) = 0
    editex(i; 0) = editex(i − 1; 0) + d(s_{i−1}; s_i)
    editex(0; j) = editex(0; j − 1) + d(t_{j−1}; t_j )
    editex(i; j) = min[editex (i − 1; j) +
                d(s_{i−1}; s_i);
                ediext(i; j − 1) + d(t_{j−1}; tj);
                editex(i − 1; j − 1) + r(s_i; t_j )]
```

Figure 4: Recurrence relation for Editex edit distance

0 1 2 3 4 5 6 7 8 9

aeiouy bp ckq dt lr mn gj fpv sxz csz

Figure 5: Editex letter groups

Zobel and Dart (1996) discuss several enhancements to the Soundex and Levenshtein string matching algorithms. One enhancement is what they call "tapering." Tapering involves weighting mismatches at the beginning of the word with a higher score than mismatches towards the end of the word. The other enhancement is what they call *phonometric methods*, in which the input strings are mapped to pronunciation based phonemic representations. The edit distance algorithm is then applied to the phonemic representations of the strings.

Zobel and Dart report that the Editex algorithm performed significantly better than alternatives they tested, including Soundex, Levenshtein edit distance, algorithms based on counting common n-gram sequences, and about ten permutations of tapering and phoneme based enhancements to assorted combinations of Soundex, n-gram counting and Levenshtein.

2.4 SecondString

SecondString, described by Cohen, Ravikumar and Fienberg (2003) is an open-source library of string-matching algorithms implemented in Java. It is freely available at the web site http://secondstring.sourceforge.net.

The SecondString library offers a wide assortment of string matching algorithms, both those based on the "edit distance" algorithm, and those based on other string matching algorithms. SecondString also provides tools for combining matching algorithms to produce hybrid-matching algorithms, tools for training on string matching metrics and tools for matching on tokens within strings for multi-token strings.

3 Baseline task

An initial set of identical names in English and Arabic script were obtained from 106 Arabic texts and 105 English texts in a corpus of newswire articles. We extracted 408 names from the English language articles and 255 names from the Arabic language articles. Manual cross-script matching identified 29 names common to both lists.

For a baseline measure, we matched the entire list of names from the Arabic language texts against the entire list of English language names using algorithms from the *SecondString* toolkit. The Arabic names were transliterated using the computer program *Artrans* produced by Basis (2004).

For each of these string matching metrics, the matching threshold was empirically set to a value that would return some matches, but minimized false matches. The Levenshtein "edit-distance" algorithm returns a simple integer indicating the number of edits required to make the two strings match. We normalized this number by using the formula $1-\left(\frac{Levenshtein(s,t)}{|s|+|t|}\right)$, where any pair of strings with a fuzzy match score less than 0.875 was not considered to be a match. The intent of dividing by the length of both names is to minimize the weight of a mismatched character in longer strings.

For the purposes of defining recall and precision, we ignored all issues dealing with the fact that many English names correctly matched more than one Arabic name, and that many Arabic names correctly matched more than one English name. The number of correct matches is the number of correct matches for each Arabic name, summed across all Arabic names having one or more matches. Recall *R* is defined as the number of correctly matched English names divided by the number of available correct English matches in the test set. Precision *P* is defined as the total number of correct names returned by the algorithm divided by the total number of names returned. The F-score is $2\cdot\frac{(PR)}{P+R}$.

Figure 5 shows the results obtained from the four algorithms that were tested. *Smith-Waterman* is based on Levenshtein edit-distance algorithm, with some parameterization of the gap score. *SLIM* is an iterative statistical learning algorithm based on a variety of estimation-maximization in which a Levenshtein edit-distance matrix is iteratively processed to find the statistical probabilities of the overlap between two strings. *Jaro* is a type n-gram algorithm which measures the number and the order of the common characters between two strings. *Needleman-Wunsch* from Cohen et al.'s (2003) *SecondString* Java code library is the Java implementation referred to as "Levenshtein edit

distance" in this report. The Levenshtein algorithms clearly out performed the other metrics.

Algorithm	Recall	Precision	F-score
Smith Waterman	14/29	14/18	0.5957
SLIM	3/29	3/8	0.1622
Jaro	8/29	8/11	0.4
NeedlemanWunsch	19/29	19/23	0.7308

Figure 5: Comparison of string similarity metrics

4 Motivation of enhancements

One insight is that each letter in an Arabic name has more than one possible letter in its English representation. For instance, the first letter of former Egyptian president Gamal Abd Al-Nasser's first name is written with the Arabic letter جـ, which in most other dialects of Arabic is pronounced either as [δZ] or [Z], most closely resembling the English pronunciation of the letter "j". As previously noted, قـ has the received pronunciation of [q], but in many dialects it is pronounced as [g], just like the Egyptian pronunciation of Nasser's first name Gamal. The conclusion is that there is no principled way to predict a single representation in English for an Arabic letter.

Similarly, Arabic representations of non-native names are not entirely predictable. Accented syllables will be given a long vowel, but in longer names, different writers will place the long vowels showing the accented syllables in different places. We observed six different ways to represent the name Milosevic in Arabic.

The full set of insights and "real-world" knowledge of the craft for representing foreign names in Arabic and English is summarized in figure 6. These rules are based on first author Dr. Andrew Freeman's[1] experience with reading and translating Arabic language texts for more than 16 years.

1)	The hamza (ء) and the 'ayn (ع) will often appear in English language texts as an apostrophe or as the vowel that follows.
2)	Names not native to Arabic will have a long vowel or diphthong for accented syllables represented by "w," "y" or "A.
3)	The high front un-rounded diphthong ("i," "ay", "igh") found in non-Arabic names will often be represented with an alif-yaa (ايـ) sequence in the Arabic script.
4)	The back rounded diphthongs, (ow, au, oo) will be represented with a single "waw" in Arabic.
5)	The Roman scripts letters "p" and "v" are represented by "b" and "f" in Arabic. The English letter "x" will appear as the sequence "ks" in Arabic
6)	Silent letters, such as final "e" and internal "gh" in English names will not appear in the Arabic script.
7)	Doubled English letters will not be represented in the Arabic script.
8)	Many Arabic names will not have any short vowels represented.
9)	The "ch" in the English name "Richard" will be represented with the two character sequence "t" (ت) and "sh" (ش). The name "Buchanan" will be represented in Arabic with the letter "k" (ك).

Figure 6: Rules for Arabic and English representations

[1] Dr. Freeman's PhD dissertation was on Arabic dialectology.

5 Implementation of the enhancements

5.1 Character Equivalence Classes (CEQ):

The implementation of the enhancements has six parts. We replaced the comparison for the character match in the Levenshtein algorithm with a function Ar($s_{i,}$ t_j) that returns zero if the character t_j from the English string is in the match set for the Arabic character $s_{i;}$, otherwise it returns a one.

```
for each i from 0 to |s|
    for each j from 0 to |t|
        levenshtein(0; 0) = 0
        levenshtein(i; 0) = i
        levenshtein(0;j) = j
        levenshtein (i;j) =
        min
          [levenshtein (i − 1; j) + 1;
          levenshtein(i; j − 1) + 1;
          levenshtein(i − 1; j − 1) + Ar(s_i; t_j )]
```

Figure 7: Cross linguistic Levenshtein

String similarity measures require the strings to have the same character set, and we chose to use transliterated Arabic so that investigators who could not read Arabic script could still view and understand the results. The full set of transliterated Arabic equivalence classes is shown in Figure 8. The set was intentionally designed to handle Arabic text transliterated into either the Buckwalter

transliteration (Buckwalter, 2002) or the default setting of the transliteration software developed by Basis Technology (Basis, 2004).

5.2 Normalizing the Arabic string

The settings used with the Basis Artrans transliteration tool transforms certain Arabic letters into English digraphs with the appropriate two characters from the following set: (kh, sh, th, dh). The Buckwalter transliteration method requires a one-to-one and recoverable mapping from the Arabic script to the transliterated script. We transformed these characters into the Basis representation with regular expressions. These regular expressions are shown in figure 9 as perl script.

Translit-eration	English equivalence class	Arabic letter
'	',a ,A,e,E,i,I,o,O,u,U	ء
|	',a ,A,e ,E,i ,I,o ,O,u ,U	آ
>	',a ,A,e ,E,i ,I,o ,O,u ,U	أ
&	',a ,A,e ,E,i ,I,o ,O,u ,U	ؤ
<	',a ,A,e ,E,i ,I,o ,O,u ,U	إ
}	',a ,A,e ,E,i ,I,o ,O,u ,U	ئ
A	',a ,A,e ,E,i ,I,o ,O,u ,U	ا
b	b ,B,p ,P,v,V	ب
p	a ,e	ة
+	a ,e	ة
t	t,T	ت
v	t ,T	ث
j	j,J,g,G	ج
H	h, H	ح
x	k, K	خ
d	d, D	د
*	d, D	ذ
r	r, R	ر
z	z, Z	ز
s	s, S,c, C	س
$	s, S	ش
S	s, S	ص
D	d, D	ض
T	t, T	ط
Z	z, Z,d, D	ظ
E	',`,c,a,A,e,E,i,I,o,O,u,U	ع
`	',`,c,a,A,e,E,i,I,o,O,u,U	ع
g	g, G	غ
f	f, F,v, V	ف
q	q, Q, g, G,k, K	ق
k	k, K,c, C,S, s	ك
l	l, L	ل
m	m, M	م
n	n, N	ن
h	h, H	هـ
w	w, W,u, u,o, O, 0	و
y	y, Y, i, I, e, E, ,j, J	ي
Y	a, A,e, E,i, I, o,O,u, U	ى
a	a, e	َ
i	i, e	ِ
u	u, o	ُ

Figure 8: Arabic to English character equivalence sets

```
$s1 =~ s/\$/sh/g;     #  normalize Buckwalter
$s1 =~ s/v/th/g;      # normalize Buckwalter
$s1 =~ s/\*/dh/g;     # normalize Buckwalter
$s1 =~ s/x/kh/g;      # normalize Buckwalter
$s1 =~ s/(F|K|N|o|~)//g; #  remove case vowels,
                      # the shadda and the sukuun
$s1 =~ s/\'aa/\|/g;   # normalize basis w/
                      # Buckwalter madda
$s1 =~ s/(U|W|I|A)/A/g; # normalize  hamza
$s1 =~ s/_//;         # eliminate underscores
$s1 =~ s/\s//g;       # eliminate white space
```

Figure 9. Normalizing the Arabic

5.3 Normalizing the English string

Normalization enhancements were aimed at making the English string more closely match the transliterated form of the Arabic string. These correspond to points 2 through 7 of the list in Figure 6. The perl code that implemented these transformations is shown in figure 10.

```
$s2 =~ s/(a|e|i|A|E|I)(e|i|y)/y/g;
        # hi dipthongs go to y in Arabic
$s2 =~ s/(e|a|o)(u|w|o)/w/g;
        # lo dipthongs go to w in Arabic
$s2 =~ s/(P|p)h/f/g;          # ph -> f in Arabic
$s2 =~ s/(S|s)ch/sh/g;        # sch is sh
$s2 =~ s/(C|c)h/tsh/g;        # ch is tsh or k ,
        # we catch the "k" on the pass
$s2 =~ s/-//g;       # eliminate all hyphens
$s2 =~ s/x/ks/g;   # x->ks in Arabic
$s2 =~ s/e( |$)/$1/g; # the silent final e
$s2 =~ s/(\S)\1/$1/g; # eliminate duplicates
$s2 =~ s/(\S)gh/$1/g; # eliminate silent gh
$s2 =~ s/\s//g;       # eliminate white space
$s2 =~ s/(\.|,|;)//g; # eliminate punctuation
```

Figure 10. Normalizing the English

5.4 Normalizing the vowel representations

Normalization of the vowel representations is based on two observations that correspond to points 2 and 8 of Figure 6. Figure 11 shows some English names represented in Arabic transliterated using the Buckwalter transliteration method.

Name in English	Name in Arabic	Arabic transliteration
Bill Clinton	بيل كلينتون	byl klyntwn
Colin Powell	كولين باول	kwlyn bAwl

Richard Cheney	ريتشارد تشيني	rytshArd tshyny

Figure 11. English names as represented in Arabic

All full, accented vowels are represented in the Arabic as a long vowel or diphthong. This vowel or diphthong will appear in the transliterated unvocalized text as either a "w," "y" or "A." Unaccented short vowels such as the "e" found in the second syllable of "Powell" are not represented in Arabic. Contrast figure 11 with the data in figure 12.

Name in Arabic	Arabic transliteration	Name in English
مصطفى الشيخ ديب	mSTfY Alshykh dyb	Mustafa al Sheikh Deeb
محمد عاطف	mHmd EATf	Muhammad Atef
حسني مبارك	Hsny mbArk	Hosni Mubarak

Figure 12. Arabic names as represented in English

The Arabic only has the lengtheners "y", "w", or "A" where there are lexically determined long vowels or diphthongs in Arabic. The English representation of these names must contain a vowel for every syllable. The edit-distance score for matching "Muhammad" with "mHmd" will fail since only 4 out of 7 characters match. Lowering the match threshold will raise the recall score while lowering the precision score. Stripping all vowels from both strings will raise the precision on the matches for Arabic names in English, but will lower the precision for English names in Arabic.

For each i from 0 to min(|*Estring*|, |*Astring*|),
each j from 0 to min(|*Estring*|, |*Astring*|)
if $Astring_i$ equals $Estring_j$
$Outstring_i = Estring_i$ increment i and j
if vowel($Astring_i$) and vowel($Estring_j$)
$Outstring_i = Estring_i$ increment i and j
if not vowel($Astring_i$) and vowel($Estring_j$)
increment j but not i
if j < |*Estring*|
$Outstring_i = Estring_i$ increment i and j
otherwise
$Outstring_i = Estring_{i,}$ increment i and j
Finally if there is anything left of *Estring*,
strip all vowels from what is left
append *Estring* to end of *Outstring*

Figure 13. Algorithm for retaining matching vowels

The algorithm presented in figure 13 retains only those vowels that are represented in both strings. The algorithm is a variant of a sorted file merge.

5.5 Normalizing "ch" representations with a separate pass

This enhancement requires a separate pass. The name "Buchanan" is represented in Arabic as "bywkAnAn" and "Richard" is "rytshArd." Thus, whichever choice the software makes for the correct value of the English substring "ch," it will choose incorrectly some significant number of times. In one pass, every "ch" in the English string gets mapped to "tsh." In a separate pass, every "ch" in the English string is transformed into a "k."

5.6 Light Stemming

The light stemming performed here was to remove the first letter of the transliterated Arabic name if it matched the prefixes "b," "l" or "w" and run the algorithm another time if the match score was below the match threshold but above another lower threshold. The first two items are prepositions that attach to any noun. The third is a conjunction that attaches to any word. Full stemming for Arabic is a separate and non-trivial problem.

6 Results

The algorithm with all enhancements was implemented in perl and in Java. Figure 14 presents the results of the enhanced algorithm on the original baseline as compared with the baseline algorithm. The enhancements improved the F-score by 22%.

Algorithm	Recall	Precision	F-score
Baseline	19/29	19/23	0.7308
Enhancements	29/29	29/32	0.9508

Figure 14. Enhanced edit distance on original data set

6.1 Results with a larger data set

After trying the algorithm out on a couple more "toy" data sets with similar results, we used a more realistic data set, which I will call the *TDT data set*. This data set was composed of 577 Arabic names and 968 English names that had been manually extracted from approximately 250 Arabic and English news articles on common topics in a NIST TDT corpus. There are 272 common names. The number of strings on the English side that correctly

match an Arabic language string is 591. The actual number of matches in the set is 641, since many Arabic strings match to the same set of English names. For instance, "Edmond Pope" has nine variants in English and six variants in Arabic. This gives 36 correct matches for the six Arabic spellings of Edmond Pope.

We varied the match threshold for various combinations of the described enhancements. The plots of the F-score, precision and recall from these experiments using the *TDT data set* are shown in figures 15, 16, and 17.

7 Discussion

Figure 15 shows that simply adding the "character equivalency classes" (CEQ) to the baseline algorithm boosts the F-score from around 48% to around 72%. Adding all other enhancements to the baseline algorithm, without adding CEQ only improves the f-score marginally. Combining these same enhancements with the CEQ raises the f-score by roughly 7% to almost 80%.

When including CEQ, the algorithm has a peak performance with a threshold near 85%. When CEQ is not included, the algorithm has a peak performance when the match threshold is around 70%. The baseline algorithm will declare that the strings match at a cutoff of 70%. Because we are normalizing by dividing by the lengths of both strings, this allows strings to match when half of their letters do not match. The CEQ forces a structure onto which characters are an allowable mismatch before the threshold is applied. This apparently leads to a reduction in the number allowable mismatches when the match threshold is tested.

The time and space complexity of the baseline Levenshtein algorithm is a function of the length of the two input strings, being |s| * |t|. This makes the time complexity (N^2) where N is the size of the average input string. The enhancements described here add to the time complexity. The increase is an average two or three extra compares per character and thus can be factored out of any equation. The new time complexity is K(|s|*|t|) where K >= 3.

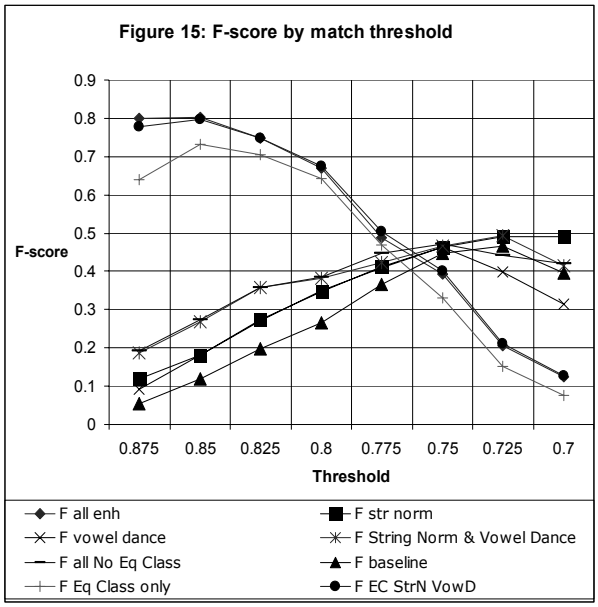
What we do here is the opposite of the approach taken by the Soundex and Editex algorithms. They try to reduce the complexity by collapsing groups of characters into a single super-class of characters. The algorithm here does some of that with the steps that normalize the strings. However, the largest boost in performance is with CEQ, which expands the number of allowable cross-language matches for many characters.

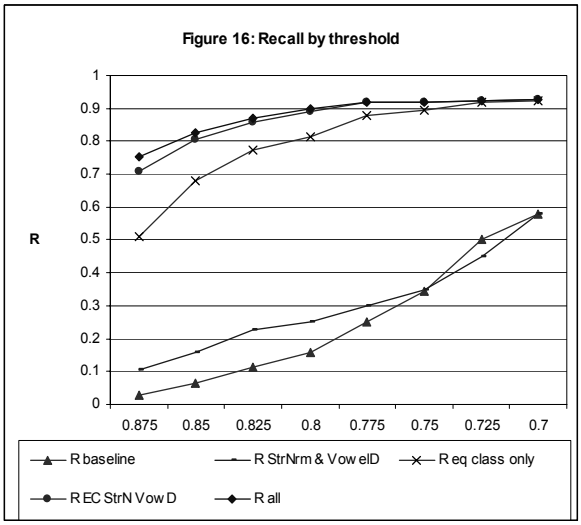
One could expect that increasing the allowable number of matches would over-generate, raising the recall while lowering the precision.

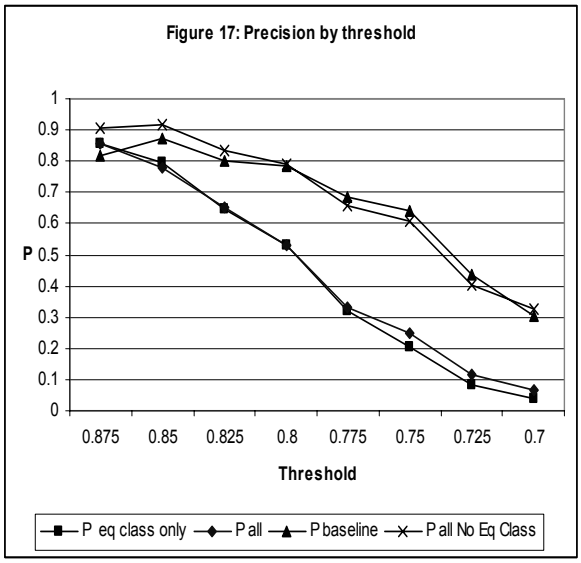
Referring to Figure 8, we see that's ome Arabic graphemes map to overlapping sets of characters in the English language strings.

Arabic جـ can be realized, as either [j] or [g], and one of the reflexes in English for Arabic ق can be [g] as well. How do we differentiate the one from the other? Quite simply, the Arabic input is not random data. Those dialects that produce ق as a [g] will as a rule not produce جـ as [g] and vice versa. The Arabic pronunciation of the string determines the correct alternation of the two characters for us *as it is written in English*. On a string-by-string basis, it is very unlikely that the two representations will conflict. The numbers show that by adding CEQ, the baseline algorithm's recall at threshold of 72.5%, goes from 57% to around 67% at a threshold of 85% for Arabic to English cross-linguistic name matching. Combining all of the enhancements raises the recall at a threshold of 85%, to 82%. As previously noted, augmenting the baseline algorithm with all enhancements except CEQ, does improve the performance dramatically. CEQ combines well with the other enhancements.

It is true that there is room for a lot improvement with an f-score of 80%. However, anyone doing cross-linguistic name matches would probably benefit by implementing some form of the character equivalence classes detailed here.







References

Basis Technology. 2004. Arabic Transliteration Module. Artrans documentation. (The documentation is available for download at http://www.basistech.com/arabic-editor.)

Bilenko, Mikael, Mooney, Ray, Cohen, William W., Ravikumar, Pradeep and Fienberg, Steve. 2003. Adaptive Name-Matching. in *Information Integration in IEEE Intelligent Systems,* 18(5): 16-23.

Buckwalter, Tim. 2002. Arabic Transliteration. http://www.qamus.org/transliteration.htm.

Cohen, William W., Ravikumar, Pradeep and Fienberg, Steve. 2003. A Comparison of String Distance Metrics for Name-Matching Tasks. *IIWeb 2003: 73-78*.

Jackson, Peter and Moulinier, Isabelle. 2002 . *Natural Language Processing for Online Applications: Text Retrieval, Extraction, and Categorization (Natural Language Processing, 5)*. John Benjamins Publishing.

Jurafsky, Daniel, and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. Prentice-Hall.

Knuth, Donald E. 1973. *The Art of Computer Programming, Volume 3: Sorting and Searching*. . Addison-Wesley Publishing Company,

Ukonnen, E. 1992. Approximate string-matching with q-grams and maximal matches. Theoretical Computer Science, 92: 191-211.

Wright, W. 1967. *A Grammar of the Arabic Language*. Cambridge. Cambridge University Press.

Zobel, Justin and Dart, Philip. 1996. Phonetic string matching: Lessons from information retrieval.in *Proceedings of the Eighteenth ACM SIGIR International Conference on Research and Development in Information Retrieval*, Zurich, Switzerland, August 1996, pp. 166-173.

Language Model-Based Document Clustering Using Random Walks

Güneş Erkan
Department of EECS
University of Michigan
Ann Arbor, MI 48109-2121
gerkan@umich.edu

Abstract

We propose a new document vector representation specifically designed for the document clustering task. Instead of the traditional term-based vectors, a document is represented as an n-dimensional vector, where n is the number of documents in the cluster. The value at each dimension of the vector is closely related to the generation probability based on the language model of the corresponding document. Inspired by the recent graph-based NLP methods, we reinforce the generation probabilities by iterating random walks on the underlying graph representation. Experiments with k-means and hierarchical clustering algorithms show significant improvements over the alternative $tf \cdot idf$ vector representation.

1 Introduction

Document clustering is one of the oldest and most studied problems of information retrieval (van Rijsbergen, 1979). Almost all document clustering approaches to date have represented documents as vectors in a *bag-of-words vector space model*, where each dimension of a document vector corresponds to a term in the corpus (Salton and McGill, 1983). General clustering algorithms are then applied to these vectors to cluster the given corpus. There have been attempts to use bigrams or even higher-order n-grams to represent documents in text categorization, the supervised counterpart of document clustering, with little success (Caropreso et al., 2001; Tan et al., 2002).

Clustering can be viewed as partitioning a set of data objects into groups such that the similarities between the objects in a same group is high while inter-group similarities are weaker. The fundamental assumption in this work is that *the documents that are likely to have been generated from similar language models are likely to be in the same cluster*. Under this assumption, we propose a new representation for document vectors specifically designed for clustering purposes.

Given a corpus, we are interested in the generation probabilities of a document based on the language models induced by other documents in the corpus. Using these probabilities, we propose a vector representation where each dimension of a document vector corresponds to a document in the corpus instead of a term in the classical representation. In other words, our document vectors are n-dimensional, where n is the number of documents in the corpus to be clustered. For the vector $\mathbf{v}_{d_i}$ of document d_i, the j^{th} element of $\mathbf{v_{d_i}}$ is closely related to the generation probability of d_i based on the language model induced by document d_j. The main steps of our method are as follows:

- For each ordered document pair (d_i, d_j) in a given corpus, we compute the generation probability of d_i from the language model induced by d_j making use of language-model approaches in information retrieval (Ponte and Croft, 1998).
- We represent each document by a vector of its generation probabilities based on other documents' language models. At this point, these vectors can be used in any clustering algorithm instead of the traditional term-based document vectors.
- Following (Kurland and Lee, 2005), our new document vectors are used to construct the underlying *generation graph*; the directed graph where documents are the nodes and link weights are proportional to the generation probabilities.
- We use *restricted random walk* probabilities to reinforce the generation probabilities and discover hidden relationships in the graph that are not obvious by the generation links. Our random walk model is similar to the one proposed by Harel and Kohen

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 479–486,
New York, June 2006.

(2001) for general spatial data represented as undirected graphs. We have extended their model to the directed graph case. We use new probabilities derived from random walks as the vector representation of the documents.

2 Generation Probabilities as Document Vectors

2.1 Language Models

The language modeling approach to information retrieval was first introduced by Ponte and Croft (1998) as an alternative (or an improvement) to the traditional $tf \cdot idf$ relevance models. In the language modeling framework, each document in the database defines a language model. The relevance of a document to a given query is ranked according to the generation probability of the query based on the underlying language model of the document. To induce a (unigram) language model from a document, we start with the maximum likelihood (ML) estimation of the term probabilities. For each term w that occurs in a document D, the ML estimation of w with respect to D is defined as

$$p_{ML}(w|D) = \frac{\mathrm{tf}(w, D)}{\sum_{w' \in D} \mathrm{tf}(w', D)}$$

where $\mathrm{tf}(w, D)$ is the number of occurences of term w in document D. This estimation is often smoothed based on the following general formula:

$$p(w|D) = \lambda p_{ML}(w|D) + (1 - \lambda) p_{ML}(w|Corpus)$$

where $p_{ML}(w|Corpus)$ is the ML estimation of w over an entire corpus which usually D is a member of. λ is the general smoothing parameter that takes different forms in various smoothing methods. Smoothing has two important roles (Zhai and Lafferty, 2004). First, it accounts for terms unseen in the document preventing zero probabilities. This is similar to the smoothing effect in NLP problems such as parsing. Second, smoothing has an idf-like effect that accounts for the generation probabilities of the common terms in the corpus. A common smoothing technique is to use Bayesian smoothing with the Dirichlet prior (Zhai and Lafferty, 2004; Liu and Croft, 2004):

$$\lambda = \frac{\sum_{w' \in D} \mathrm{tf}(w', D)}{\sum_{w' \in D} \mathrm{tf}(w', D) + \mu}$$

Here, μ is the smoothing parameter. Higher values of μ mean more aggressive smoothing.

Assuming the terms in a text are independent from each other, the generation probability of a text sequence S given the document D is the product of the generation probabilities of the terms of S:

$$p(S|D) = \prod_{w \in S} p(w|D) \tag{1}$$

In the context of information retrieval, S is a query usually composed of few terms. In this work, we are interested in the generation probabilities of entire documents that usually have in the order of hundreds of unique terms. If we use Equation 1, we end up having unnatural probabilities which are irrepresentably small and cause floating point underflow. More importantly, longer documents tend to have much smaller generation probabilities no matter how closely related they are to the generating language model. However, as we are interested in the generation probabilities between all pairs of documents, we want to be able to compare two different generation probabilities from a fixed language model regardless of the target document sizes. This is not a problem in the classical document retrieval setting since the given query is fixed, and generation probabilities for different queries are not compared against each other. To address these problems, following (Lavrenko et al., 2002; Kurland and Lee, 2005), we "flatten" the probabilities by normalizing them with respect to the document size:

$$p_{\text{flat}}(S|D) = p(S|D)^{\frac{1}{|S|}} \tag{2}$$

where $|S|$ is the number of terms in S. p_{flat} provides us with meaningful values which are comparable among documents of different sizes.

2.2 Using Generation Probabilities as Document Representations

Equation 2 suggests a representation of the relationship of a document with the other documents in a corpus. Given a corpus of n documents to cluster, we form an n-dimensional *generation vector* $\mathbf{g}_{d_i} = (g_{d_i 1}, g_{d_i 2}, \ldots, g_{d_i n})$ for each document d_i where

$$g_{d_i j} = \begin{cases} 0 & \text{if } i = j, \\ p_{\text{flat}}(d_i|d_j) & \text{otherwise} \end{cases} \tag{3}$$

We can use these generation vectors in any clustering algorithm we prefer instead of the classical term-based $tf \cdot idf$ vectors. The intuition behind this idea becomes clearer when we consider the underlying directed graph representation, where each document is a node and the weight of the link from d_i to d_j is equal to $p_{\text{flat}}(d_i|d_j)$. An appropriate analogy here is the citation graph of scientific papers. The generation graph can be viewed as a model where documents *cite* each other. However, unlike real citations, the generation links are weighted and automatically induced from the content.

The similarity function used in a clustering algorithm over the generation vectors becomes a measure of structural similarity of two nodes in the generation graph. Work on bibliometrics uses various similarity metrics to assess the relatedness of scientific papers by looking at the citation vectors (Boyack et al., 2005). Graph-based

similarity metrics are also used to detect semantic similarity of two documents on the Web (Maguitman et al., 2005). Cosine, also the standard metric used in $tf \cdot idf$ based document clustering, is one of these metrics. Intuitively, the cosine of the citation vectors (i.e. vector of outgoing link weights) of two nodes is high when they link to similar sets of nodes with similar link weights. Hence, the cosine of two generation vectors is a measure of how likely two documents are generated from the same documents' language models.

The generation probability in Equation 2 with a smoothed language model is never zero. This creates two potential problems if we want to use the vector of Equation 3 directly in a clustering algorithm. First, we only want strong generation links to contribute in the similarity function since a low generation probability is not an evidence for semantic relatedness. This intuition is similar to throwing out the stopwords from the documents before constructing the $tf \cdot idf$ vectors to avoid coincidental similarities between documents. Second, having a dense vector with lots of non-zero elements will cause efficiency problems. Vector length is assumed to be a constant factor in analyzing the complexity of the clustering algorithms. However, our generation vectors are n-dimensional, where n is the number of documents. In other words, vector size is not a constant factor anymore, which causes a problem of scalability to large data sets. To address these problems, we use what Kurland and Lee (2005) define as *top generators*: Given a document d_i, we consider only c documents that yield the largest generation probabilities and discard others. The resultant n-dimensional vector, denoted $\mathbf{g}^c_{d_i}$, has at most c non-zero elements, which are the largest c elements of $\mathbf{g}_{d_i}$. For a given constant c, with a sparse vector representation, certain operations (e.g. cosine) on such vectors can be done in constant time independent of n.

2.3 Reinforcing Links with Random Walks

Generation probabilities are only an approximation of semantic relatedness. Using the underlying directed graph interpretation of the generation probabilities, we aim to get better approximations by accumulating the generation link information in the graph. We start with some definitions. We denote a (directed) graph as $G(V, w)$ where V is the set of nodes and $w : V \times V \rightarrow \mathbb{R}$ is the *link weight function*. We formally define a generation graph as follows:

Definition 1 *Given a corpus $\mathcal{C} = \{d_1, d_2, \ldots, d_n\}$ with n documents, and a constant c, the* generation graph *of $\mathcal{C}$ is a directed graph $G_c(\mathcal{C}, w)$, where $w(d_i, d_j) = g^c_{d_i j}$.*

Definition 2 *A t-step* random walk *on a graph $G(V, w)$ that starts at node $v_0 \in V$ is a sequence of nodes $v_0, v_1, \ldots, v_t \in V$ where $w(v_i, v_{i+1}) > 0$ for all $0 \leq i < t$. The* probability *of a t-step random walk is defined as $\prod_{i=0}^{t-1} q_{v_i v_{i+1}}$ where*

$$q_{v_i v_{i+1}} = \frac{w(v_i, v_{i+1})}{\sum_{u \in V} w(v_i, u)}$$

q_{uv} is called the transition probability *from node u to node v.*

For example, for a generation graph G_c, there are at most c 1-step random walks that start at a given node with probabilities proportional to the weights of the outgoing generation links of that node.

Suppose there are three documents A, B, and C in a generation graph. Suppose also that there are "strong" generation links from A to B and B to C, but no link from A to C. The intuition says that A must be semantically related to C to a certain degree although there is no generation link between them depending on C's language model. We approximate this relation by considering the probabilities of 2-step (or longer) random walks from A to C although there is no 1-step random walk from A to C.

Let q^t_{uv} denote the probability that an t-step random walk starts at u and ends at v. An interesting property of random walks is that for a given node v, q^∞_{uv} does not depend on u. In other words, the probability of a random walk ending up at v "in the long run" does not depend on its starting point (Seneta, 1981). This limiting probability distribution of an infinite random walk over the nodes is called the *stationary distribution* of the graph. The stationary distribution is uninteresting to us for clustering purposes since it gives an information related to the global structure of the graph. It is often used as a measure to *rank* the structural importance of the nodes in a graph (Brin and Page, 1998). For clustering, we are more interested in the local similarities inside a "cluster" of nodes that separate them from the rest of the graph. Furthermore, the generation probabilities lose their significance during long random walks since they get multiplied at each step. Therefore, we compute q^t for small values of t. Finally, we define the following:

Definition 3 *The t-*step generation probability *of document d_i from the language model of d_j:*

$$gen^t(d_i|d_j) = \frac{\sum_{s=1}^{t} q^s_{d_i d_j}}{t}$$

*$\mathbf{gen}^t_{d_i} = (gen^t(d_i|d_1), gen^t(d_i|d_2), \ldots, gen^t(d_i|d_n))$ is the t-*step generation vector *of document d_i. We will often write $\mathbf{gen}^t$ omitting the document name when we are not talking about the vector of a specific document.*

$\text{gen}_t(d_i, d_j)$ is a measure of how likely a random walk that starts at d_i will visit d_j in t or fewer steps. It helps us to discover "hidden" similarities between documents

that are not immediately obvious from 1-step generation links. Note that when $t = 1$, $\mathbf{gen}^1_{d_i}$ is nothing but $\mathbf{g}^c_{d_i}$ normalized such that the sum of the elements of the vector is 1. The two are practically the same representations since we compute the cosine of the vectors during clustering.

3 Related Work

Our work is inspired by three main areas of research. First, the success of language modeling approaches to information retrieval (Ponte and Croft, 1998) is encouraging for a similar twist to document representation for clustering purposes. Second, graph-based inference techniques to discover "hidden" textual relationships like the one we explored in our random walk model have been successfully applied to other NLP problems such as summarization (Erkan and Radev, 2004; Mihalcea and Tarau, 2004; Zha, 2002), prepositional phrase attachment (Toutanova et al., 2004), and word sense disambiguation (Mihalcea, 2005). Unlike our approach, these methods try to exploit the global structure of a graph to *rank* the nodes of the graph. For example, Erkan and Radev (2004) find the stationary distribution of the random walk on a graph of sentences to rank the salience scores of the sentences for extractive summarization. Their link weight function is based on cosine similarity. Our graph construction based on generation probabilities is inherited from (Kurland and Lee, 2005), where authors used a similar generation graph to rerank the documents returned by a retrieval system based on the stationary distribution of the graph. Finally, previous research on clustering graphs with restricted random walks inspired us to cluster the generation graph using a similar approach. Our t-step random walk approach is similar to the one proposed by Harel and Koren (2001). However, their algorithm is proposed for "spatial data" where the nodes of the graph are connected by undirected links that are determined by a (symmetric) similarity function. Our contribution in this paper is to use their approach on textual data by using generation links, and extend the method to directed graphs.

There is an extensive amount of research on document clustering or clustering algorithms in general that we can not possibly review here. After all, we do not present a new clustering algorithm, but rather a new representation of textual data. We explain some popular clustering algorithms and evaluate our representation using them in Section 4. Few methods have been proposed to cluster documents using a representation other than the traditional $tf \cdot idf$ vector space (or similar term-based vectors). Using a bipartite graph of terms and documents and then clustering this graph based on spectral methods is one of them (Dhillon, 2001; Zha et al., 2001). There are also general spectral methods that start with $tf \cdot idf$ vectors, then map them to a new space with fewer dimensions before initiating the clustering algorithm (Ng et al., 2001).

The *information-theoretic* clustering algorithms are relevant to our framework in the sense that they involve probability distributions over words just like the language models. However, instead of looking at the word distributions at the individual document level, they make use of the *joint* distribution of words and documents. For example, given the set of documents X and the set of words Y in the document collection, Slonim and Tishby (2000) first try to find a word clustering $\hat{Y}$ such that the mutual information $I(Y, \hat{Y})$ is minimized (for good compression) while maximizing the $I(\hat{Y}, X)$ (for preserving the original information). Then the same procedure is used for clustering documents using the word clusters from the first step. Dhillon et. al. (2003) propose a *co-clustering* version of this information-theoretic method where they cluster the words and the documents concurrently.

4 Evaluation

We evaluated our new vector representation by comparing it against the traditional $tf \cdot idf$ vector space representation. We ran k-means, single-link, average-link, and complete-link clustering algorithms on various data sets using both representations. These algorithms are among the most popular ones that are used in document clustering.

4.1 General Experimental Setting

Given a corpus, we stemmed all the documents, removed the stopwords and constructed the $tf \cdot idf$ vector for each document by using the *bow toolkit* (McCallum, 1996). We computed the idf of each term using the following formula:

$$\text{idf}(w) = log_2 \left(\frac{n}{\text{df}(w)} \right)$$

where n is the total number of documents and $\text{df}(w)$ is the number of documents that the term w appears in.

We computed flattened generation probabilities (Equation 2) for all ordered pairs of documents in a corpus, and then constructed the corresponding generation graph (Definition 1). We used Dirichlet-smoothed language models with the smoothing parameter $\mu = 1000$, which can be considered as a typical value used in information retrieval. While computing the generation link vectors, we did not perform extensive parameter tuning at any stage of our method. However, we observed the following:

- When c (number of outgoing links per document) was very small (less than 10), our methods performed poorly. This is expected with such a sparse vector representation for documents. However, the performance got rapidly and almost monotonically

better as we increased c until around $c = 80$, where the performance stabilized and dropped after around $c = 100$. We conclude that using bounded number of outgoing links per document is not only more efficient but also necessary as we motivated in Section 2.2.

- We got the best results when the random walk parameter $t = 3$. When $t > 3$, the random walk goes "out of the cluster" and $\mathbf{gen}^t$ vectors become very dense. In other words, almost all of the graph is reachable from a given node with 4-step or longer random walks (assuming c is around 80), which is an indication of a "small world" effect in generation graphs (Watts and Strogatz, 1998).

Under these observations, we will only report results using vectors $\mathbf{gen}^1$, $\mathbf{gen}^2$ and $\mathbf{gen}^3$ with $c = 80$ regardless of the data set and the clustering algorithm.

4.2 Experiments with k-means

4.2.1 Algorithm

k-means is a clustering algorithm popular for its simplicity and efficiency. It requires k, the number of clusters, as input, and partitions the data set into exactly k clusters. We used a version of k-means that uses cosine similarity to compute the distance between the vectors. The algorithm can be summarized as follows:

1. randomly select k document vectors as the initial cluster centroids;
2. assign each document to the cluster whose centroid yields the highest cosine similarity;
3. recompute the centroid of each cluster. (centroid vector of a cluster is the average of the vectors in that cluster);
4. stop if none of the centroid vectors has changed at step 3. otherwise go to step 2.

4.2.2 Data

k-means is known to work better on data sets in which the documents are nearly evenly distributed among different clusters. For this reason, we tried to pick such corpora for this experiment to be able to get a fair comparison between different document representations. The first corpus we used is *classic3*,[1] which is a collection of technical paper abstracts in three different areas. We used two corpora, *bbc* and *bbcsport*, that are composed of BBC news articles in general and sports news, respectively. [2] Both corpora have 5 news classes each. *20news*[3] is a corpus of newsgroup articles composed of 20 classes. Table 1 summarizes the corpora we used together with the sizes of the smallest and largest class in each of them.

Corpus	Documents	Classes	Smallest	Largest
classic3	3891	3	1033	1460
bbcsport	737	5	100	265
bbc	2225	5	386	511
20news	18846	20	628	999

Table 1: The corpora used in the k-means experiments.

4.2.3 Results

We used two different metrics to evaluate the results of the k-means algorithm; accuracy and mutual information. Let l_i be the label assigned to d_i by the clustering algorithm, and α_i be d_i's actual label in the corpus. Then,

$$\text{Accuracy} = \frac{\sum_{i=1}^{n} \delta(\text{map}(l_i), \alpha_i)}{n}$$

where $\delta(x, y)$ equals 1 if $x = y$ and equals zero otherwise. $\text{map}(l_i)$ is the function that maps the output label set of the k-means algorithm to the actual label set of the corpus. Given the confusion matrix of the output, best such mapping function can be efficiently found by Munkres's algorithm (Munkres, 1957).

Mutual information is a metric that does not require a mapping function. Let $L = \{l_1, l_2, \ldots, l_k\}$ be the output label set of the k-means algorithm, and $A = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ be the actual label set of the corpus with the underlying assignments of documents to these sets. Mutual information (MI) of these two labelings is defined as:

$$\text{MI}(L, A) = \sum_{l_i \in L, \alpha_j \in A} P(l_i, \alpha_j) \cdot \log_2 \frac{P(l_i, \alpha_j)}{P(l_i) \cdot P(\alpha_j)}$$

where $P(l_i)$ and $P(\alpha_j)$ are the probabilities that a document is labeled as l_i and α_j by the algorithm and in the actual corpus, respectively; $P(l_i, \alpha_j)$ is the probability that these two events occur at the same time. These values can be derived from the confusion matrix. We map the MI metric to the $[0, 1]$ interval by normalizing it with the maximum possible MI that can be achieved with the corpus. Normalized MI is defined as

$$\widehat{\text{MI}} = \frac{\text{MI}(L, A)}{\text{MI}(A, A)}$$

[1] `ftp://ftp.cs.cornell.edu/pub/smart`

[2] `http://www.cs.tcd.ie/Derek.Greene/research/datasets.html` BBC corpora came in preprocessed format so that we did not perform the processing with the *bow* toolkit mentioned in Section 4.1

[3] `http://people.csail.mit.edu/jrennie/20Newsgroups`

One disadvantage of k-means is that its performance is very dependent on the initial selection of cluster centroids. Two approaches are usually used when reporting the performance of k-means. The algorithm is run multiple times; then either the average performance of these runs or the best performance achieved is reported. Reporting the best performance is not very realistic since we would not be clustering a corpus if we already knew the class labels. Reporting the average may not be very informative since the variance of multiple runs is usually large. We adopt an approach that is somewhere in between. We use "true seeds" to initialize k-means, that is, we *randomly* select k document vectors *that belong to each of the true classes* as the initial centroids. This is not an unrealistic assumption since we initially know the number of classes, k, in the corpus, and the cost of finding one example document from each class is not usually high. This way, we also aim to reduce the variance of the performance of different runs for a better analysis.

Table 2 shows the results of k-means algorithm using $tf \cdot idf$ vectors versus generation vectors $\mathbf{gen}^1$ (plain flattened generation probabilities), $\mathbf{gen}^2$ (2-step random walks), $\mathbf{gen}^3$ (3-step random walks). Taking advantage of the relatively larger size and number of classes of *20news* corpus, we randomly divided it into disjoint partitions with 4, 5, and 10 classes which provided us with 5, 4, and 2 new corpora, respectively. We named them *4news-1*, *4news-2*, ..., *10news-2* for clarity. We ran k-means with 30 distinct initial seed sets for each corpus.

The first observation we draw from Table 2 is that even $\mathbf{gen}^1$ vectors perform better than the $tfidf$ model. This is particularly surprising given that $\mathbf{gen}^1$ vectors are sparser than the $tf \cdot idf$ representation for most documents.[4] All $\mathbf{gen}^t$ vectors clearly outperform $tf \cdot idf$ model often by a wide margin. The performance also gets better (not always significantly though) in almost all data sets as we increase the random walk length, which indicates that random walks are useful in reinforcing generation links and inducing new relationships. Another interesting observation is that the confidence intervals are also narrower for generation vectors, and tend to get even narrower as we increase t.

4.3 Experiments with Hierarchical Clustering

4.3.1 Algorithms

Hierarchical clustering algorithms start with the trivial clustering of the corpus where each document defines a separate cluster by itself. At each iteration, two "most similar" separate clusters are merged. The algorithm stops after $n-1$ iterations when all the documents are merged into a single cluster.

Hierarchical clustering algorithms differ in how they define the similarity between two clusters at each merging step. We experimented with three of the most popular algorithms using cosine as the similarity metric between two vectors. *Single-link clustering* merges two clusters whose most similar members have the highest similarity. *Complete-link clustering* merges two clusters whose least similar members have the highest similarity. *Average-link clustering* merges two clusters that yield the highest average similarity between all pairs of documents.

4.3.2 Data

Corpus	Documents	Classes	Smallest	Largest
Reuters	8646	57	2	3735
TDT2	10160	87	2	1843

Table 3: The corpora used in the hierarchical clustering experiments.

Although hierarchical algorithms are not very efficient, they are useful when the documents are not evenly distributed among the classes in the corpus and some classes exhibit a "hierarchical" nature; that is, some classes in the data might be semantically overlapping or they might be in a subset/superset relation with each other. We picked two corpora that may exhibit such nature to a certain extent. Reuters-21578[5] is a collection of news articles from Reuters. TDT2[6] is a similar corpus of news articles collected from six news agencies in 1998. They contain documents labeled with zero, one or more class labels. For each corpus, we used only the documents with exactly one label. We also eliminated classes with only one document since clustering such classes is trivial. We ended up with two collections summarized in Table 3.

4.3.3 Results

The output of a hierarchical clustering algorithm is a *tree* where leaves are the documents and each node in the tree shows a cluster merging operation. Therefore each subtree represents a cluster. We assume that each class of documents in the corpus form a cluster subtree at some point during the construction of the tree. To evaluate the cluster tree, we use F-measure proposed in (Larsen and Aone, 1999). F-measure for a class c_i in the corpus and a subtree s_j is defined as

$$F(c_i, s_j) = \frac{2 \cdot R(c_i, s_j) \cdot P(c_i, s_j)}{R(c_i, s_j) + P(c_i, s_j)}$$

[4] Remember that we set $c = 80$ in our experiments which means that there can be a maximum of 80 non-zero elements in $\mathbf{gen}^1$. Most documents have more than 80 unique terms in them.

[5] http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html

[6] http://www.nist.gov/speech/tests/tdt/tdt98/index.htm

Corpus	k	Accuracy ($\times 100$)				Normalized Mutual Information ($\times 100$)			
		$tf \cdot idf$	$\mathbf{gen}^1$	$\mathbf{gen}^2$	$\mathbf{gen}^3$	$tf \cdot idf$	$\mathbf{gen}^1$	$\mathbf{gen}^2$	$\mathbf{gen}^3$
classic3	3	95.76 ± 1.28	97.92 ± 0.69	98.70 ± 0.19	$\mathbf{98.79 \pm 0.03}$	84.69 ± 2.50	91.16 ± 1.90	93.39 ± 0.59	$\mathbf{93.64 \pm 0.12}$
4news-1	4	72.32 ± 2.80	82.76 ± 1.55	85.86 ± 1.30	$\mathbf{86.58 \pm 1.23}$	49.85 ± 3.39	64.85 ± 1.56	69.72 ± 0.88	$\mathbf{70.73 \pm 0.89}$
4news-2	4	63.42 ± 2.83	77.33 ± 2.14	80.66 ± 1.92	$\mathbf{81.29 \pm 1.80}$	34.02 ± 2.97	54.55 ± 1.90	59.50 ± 1.45	$\mathbf{60.46 \pm 1.31}$
4news-3	4	62.35 ± 2.51	74.96 ± 3.35	78.60 ± 3.67	$\mathbf{78.85 \pm 3.75}$	33.74 ± 2.65	51.94 ± 3.43	58.15 ± 3.08	$\mathbf{59.24 \pm 2.95}$
4news-4	4	73.14 ± 2.05	81.36 ± 1.54	83.54 ± 1.50	$\mathbf{84.22 \pm 1.49}$	54.24 ± 2.74	66.47 ± 1.24	69.78 ± 0.90	$\mathbf{70.41 \pm 0.83}$
4news-5	4	69.43 ± 3.68	87.06 ± 2.33	$\mathbf{90.07 \pm 1.66}$	90.06 ± 1.47	42.58 ± 3.90	66.49 ± 3.27	71.95 ± 2.45	$\mathbf{71.96 \pm 2.15}$
5news-1	5	59.00 ± 2.43	73.91 ± 2.30	76.35 ± 2.86	$\mathbf{76.75 \pm 2.58}$	36.53 ± 2.94	56.84 ± 3.19	60.74 ± 2.87	$\mathbf{61.40 \pm 2.54}$
5news-2	5	55.59 ± 2.88	69.32 ± 2.17	72.75 ± 1.74	$\mathbf{73.05 \pm 1.82}$	30.94 ± 2.45	48.00 ± 2.18	52.77 ± 1.52	$\mathbf{53.75 \pm 1.30}$
5news-3	5	71.07 ± 2.39	83.78 ± 2.16	86.14 ± 2.15	$\mathbf{86.28 \pm 2.22}$	49.65 ± 2.64	67.09 ± 2.12	71.13 ± 1.87	$\mathbf{71.70 \pm 1.67}$
5news-4	5	70.06 ± 2.64	80.20 ± 1.84	82.61 ± 1.69	$\mathbf{82.66 \pm 1.78}$	50.04 ± 2.82	63.69 ± 1.73	66.89 ± 1.32	$\mathbf{67.49 \pm 1.14}$
bbc	5	80.73 ± 2.80	87.46 ± 2.63	89.88 ± 2.57	$\mathbf{90.80 \pm 2.26}$	63.34 ± 3.23	74.19 ± 2.93	78.20 ± 2.63	$\mathbf{79.39 \pm 2.30}$
bbcsport	5	79.25 ± 2.87	94.25 ± 0.89	95.44 ± 0.42	$\mathbf{95.56 \pm 0.31}$	63.94 ± 3.27	84.59 ± 1.34	86.42 ± 0.71	$\mathbf{86.54 \pm 0.58}$
10news-1	10	50.11 ± 2.30	66.20 ± 2.12	69.18 ± 1.73	$\mathbf{69.76 \pm 1.61}$	39.98 ± 1.99	55.21 ± 1.67	58.86 ± 1.15	$\mathbf{59.70 \pm 0.96}$
10news-2	10	54.12 ± 1.76	70.01 ± 2.00	73.41 ± 1.78	$\mathbf{74.14 \pm 1.75}$	42.44 ± 1.55	60.64 ± 1.40	65.20 ± 1.11	$\mathbf{66.41 \pm 1.02}$
20news	20	41.46 ± 1.03	55.75 ± 1.25	58.97 ± 1.29	$\mathbf{60.26 \pm 0.99}$	38.28 ± 0.73	52.44 ± 0.74	56.34 ± 0.71	$\mathbf{57.47 \pm 0.53}$

Table 2: Performances of different vector representations using k-means (average of 30 runs $\pm 95\%$ confidence interval).

Algorithm	TDT2				Reuters-21578			
	$tf \cdot idf$	$\mathbf{gen}^1$	$\mathbf{gen}^2$	$\mathbf{gen}^3$	$tf \cdot idf$	$\mathbf{gen}^1$	$\mathbf{gen}^2$	$\mathbf{gen}^3$
single-link	65.25	82.96	**84.22**	83.92	59.35	59.37	65.70	**66.15**
average-link	90.78	93.53	94.04	**94.13**	78.25	79.17	77.24	**81.37**
complete-link	29.07	25.04	27.19	**34.67**	43.66	42.79	45.91	**48.36**

Table 4: Performances (F-measure $\times 100$) of different vector representations using hierarchical algorithms on two corpora.

where $R(c_i, s_j)$ and $P(c_i, s_j)$ is the recall and the precision of s_j considering the class c_i. Let S be the set of subtrees in the output cluster tree, and C be the set of classes. F-measure of the entire tree is the weighted average of the maximum F-measures of all the classes:

$$F(C, S) = \sum_{c \in C} \frac{n_c}{n} \max_{s \in S} F(c, s)$$

where n_c is the number of documents that belong to class c.

We ran all three algorithms for both corpora. Unlike k-means, hierarchical algorithms we used are deterministic. Table 4 summarizes our results. An immediate observation is that average-link clustering performs much better than other two algorithms independent of the data set or the document representation, which is consistent with earlier research (Zhao and Karypis, 2002). The highest result (shown boldface) for each algorithm and corpus was achieved by using generation vectors. However, unlike in the k-means experiments, $tf \cdot idf$ was able to outperform $\mathbf{gen}^1$ and $\mathbf{gen}^2$ in one or two cases. $\mathbf{gen}^2$ yielded the best result instead of $\mathbf{gen}^3$ in one of the six cases.

5 Conclusion

We have presented a language model inspired approach to document clustering. Our results show that even the simplest version of our approach with nearly no parameter tuning can outperform traditional $tf \cdot idf$ models by a wide margin. Random walk iterations on our graph-based model have improved our results even more. Based on the success of our model, we will investigate various graph-based relationships for explaining semantic structure of text collections in the future. Possible applications include information retrieval, text clustering/classification and summarization.

Acknowledgments

I would like to thank Dragomir Radev for his useful comments. This work was partially supported by the U.S. National Science Foundation under the following two grants: 0329043 "Probabilistic and link-based Methods for Exploiting Very Large Textual Repositories" administered through the IDM program and 0308024 "Collaborative Research: Semantic Entity and Relation Extraction from Web-Scale Text Document Collections" administered by the HLT program. All opinions, findings, conclusions, and recommendations in this paper are made by the authors and do not necessarily reflect the views of the National Science Foundation.

References

Kevin W. Boyack, Richard Klavans, and Katy Börner. 2005. Mapping the backbone of science. *Scientometrics*, 64(3):351–374.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the*

7th International World Wide Web Conference, pages 107–117.

Maria Fernanda Caropreso, Stan Matwin, and Fabrizio Sebastiani. 2001. A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. In Amita G. Chin, editor, *Text Databases and Document Management: Theory and Practice*, pages 78–102. Idea Group Publishing, Hershey, US.

Inderjit S. Dhillon, Subramanyam Mallela, and Dharmendra S. Modha. 2003. Information-theoretic co-clustering. In Pedro Domingos, Christos Faloutsos, Ted SEnator, Hillol Kargupta, and Lise Getoor, editors, *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-03)*, pages 89–98, New York, August 24–27. ACM Press.

Inderjit S. Dhillon. 2001. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the Seventh ACM SIGKDD Conference*, pages 269–274.

Güneş Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

David Harel and Yehuda Koren. 2001. Clustering spatial data using random walks. In *Proceedings of the Seventh ACM SIGKDD Conference*, pages 281–286, New York, NY, USA. ACM Press.

Oren Kurland and Lillian Lee. 2005. PageRank without hyperlinks: Structural re-ranking using links induced by language models. In *Proceedings of SIGIR*.

Bjornar Larsen and Chinatsu Aone. 1999. Fast and effective text mining using linear-time document clustering. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22, New York, NY, USA. ACM Press.

Victor Lavrenko, James Allan, Edward DeGuzman, Daniel LaFlamme, Veera Pollard, and Stephen Thomas. 2002. Relevance models for topic detection and tracking. In *Proceedings of HLT*, pages 104–110.

Xiaoyong Liu and W. Bruce Croft. 2004. Cluster-based retrieval using language models. In *Proceedings of SIGIR*, pages 186–193.

Ana G. Maguitman, Filippo Menczer, Heather Roinestad, and Alessandro Vespignani. 2005. Algorithmic detection of semantic similarity. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 107–116, New York, NY, USA. ACM Press.

Andrew Kachites McCallum. 1996. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 404–411, Barcelona, Spain, July. Association for Computational Linguistics.

Rada Mihalcea. 2005. Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 411–418, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

James Munkres. 1957. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, March.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856.

Jay M. Ponte and W. Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of SIGIR*, pages 275–281.

G. Salton and M. J. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw Hill.

E. Seneta. 1981. *Non-negative matrices and markov chains*. Springer-Verlag, New York.

Noam Slonim and Naftali Tishby. 2000. Document clustering using word clusters via the information bottleneck method. In *SIGIR*, pages 208–215.

Chade-Meng Tan, Yuan-Fang Wang, and Chan-Do Lee. 2002. The use of bigrams to enhance text categorization. *Inf. Process. Manage*, 38(4):529–546.

Kristina Toutanova, Christopher D. Manning, and Andrew Y. Ng. 2004. Learning random walk models for inducing word dependency distributions. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 103, New York, NY, USA. ACM Press.

Cornelis J. van Rijsbergen. 1979. *Information Retrieval*. Butterworths.

Duncan J. Watts and Steven H. Strogatz. 1998. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, June 4.

Hongyuan Zha, Xiaofeng He, Chris H. Q. Ding, Ming Gu, and Horst D. Simon. 2001. Bipartite graph partitioning and data clustering. In *Proceedings of CIKM*, pages 25–32.

Hongyuan Zha. 2002. Generic Summarization and Key Phrase Extraction Using Mutual Reinforcement Principle and Sentence Clustering. Tampere, Finland.

Chengxiang Zhai and John Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst. (TOIS)*, 22(2):179–214.

Ying Zhao and George Karypis. 2002. Evaluation of hierarchical clustering algorithms for document datasets. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 515–524, New York, NY, USA. ACM Press.

Unlimited vocabulary speech recognition for agglutinative languages

Mikko Kurimo[1], **Antti Puurula**[1], **Ebru Arisoy**[2], **Vesa Siivola**[1],
Teemu Hirsimäki[1], **Janne Pylkkönen**[1], **Tanel Alumäe**[3], **Murat Saraclar**[2]
[1] Adaptive Informatics Research Centre, Helsinki University of Technology
P.O.Box 5400, FIN-02015 HUT, Finland
{Mikko.Kurimo,Antti.Puurula,Vesa.Siivola}@tkk.fi
[2] Bogazici University, Electrical and Electronics Eng. Dept.
34342 Bebek, Istanbul, Turkey
{arisoyeb,murat.saraclar}@boun.edu.tr
[3] Laboratory of Phonetics and Speech Technology,
Institute of Cybernetics, Tallinn Technical University, Estonia
tanel.alumae@phon.ioc.ee

Abstract

It is practically impossible to build a word-based lexicon for speech recognition in agglutinative languages that would cover all the relevant words. The problem is that words are generally built by concatenating several prefixes and suffixes to the word roots. Together with compounding and inflections this leads to millions of different, but still frequent word forms. Due to inflections, ambiguity and other phenomena, it is also not trivial to automatically split the words into meaningful parts. Rule-based morphological analyzers can perform this splitting, but due to the handcrafted rules, they also suffer from an out-of-vocabulary problem. In this paper we apply a recently proposed fully automatic and rather language and vocabulary independent way to build subword lexica for three different agglutinative languages. We demonstrate the language portability as well by building a successful large vocabulary speech recognizer for each language and show superior recognition performance compared to the corresponding word-based reference systems.

1 Introduction

Speech recognition for dictation or prepared radio and television broadcasts has had huge advances during the last decades. For example, broadcast news (BN) in English can now be recognized with about ten percent word error rate (WER) (NIST, 2000) which results in mostly quite understandable text. Some rare and new words may be missing but the result has proven to be sufficient for many important applications, such as browsing and retrieval of recorded speech and information retrieval from the speech (Garofolo et al., 2000). However, besides the development of powerful computers and new algorithms, a crucial factor in this development is the vast amount of transcribed speech and suitable text data that has been collected for training the models. The problem faced in porting the BN recognition systems to conversational speech or to other languages is that almost as much new speech and text data have to be collected again for the new task.

The reason for the need for a vast amount of training texts is that the state-of-the-art statistical language models contain a huge amount of parameters to be estimated in order to provide a proper probability for any possible word sequence. The main reason for the huge model size is that for an acceptable coverage in an English BN task, the vocabulary must be very large, at least 50,000 words, or more. For languages with a higher degree of word inflections than English, even larger vocabularies are required. This paper focuses on the agglutinative languages in which words are frequently formed by concatenating one or more stems, prefixes, and suffixes. For these languages in which the words are often highly inflected as well as formed from several morphemes, even a vocabulary of 100,000 most common words would not give sufficient coverage (Kneissler and

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 487–494,
New York, June 2006.

Klakow, 2001; Hirsimäki et al., 2005). Thus, the solution to the language modeling clearly has to involve splitting of words into smaller modeling units that could then be adequately modeled.

This paper focuses on solving the vocabulary problem for several languages in which the speech and text database resources are much smaller than for the world's main languages. A common feature for the agglutinative languages, such as Finnish, Estonian, Hungarian and Turkish is that the large vocabulary continuous speech recognition (LVCSR) attempts so far have not resulted comparable performance to the English systems. The reason for this is not only the language modeling difficulties, but, of course, the lack of suitable speech and text training data resources. In (Geutner et al., 1998; Siivola et al., 2001) the systems aim at reducing the active vocabulary and language models to a feasible size by clustering and focusing. In (Szarvas and Furui, 2003; Alumäe, 2005; Hacioglu et al., 2003) the words are split into morphemes by language-dependent hand-crafted morphological rules. In (Kneissler and Klakow, 2001; Arisoy and Arslan, 2005) different combinations of words, grammatical morphemes and endings are utilized to decrease the OOV rate and optimize the speech recognition accuracy. However, constant large improvements over the conventional word-based language models in LVCSR have been rare.

The approach presented in this paper relies on a data-driven algorithm called Morfessor (Creutz and Lagus, 2002; Creutz and Lagus, 2005) which is a language independent unsupervised machine learning method to find morpheme-like units (called statistical morphs) from a large text corpus. This method has several advantages over the rule-based grammatical morphemes, e.g. that no hand-crafted rules are needed and all words can be processed, even the foreign ones. Even if good grammatical morphemes are available, the language modeling results by the statistical morphs seem to be at least as good, if not better (Hirsimäki et al., 2005). In this paper we evaluate the statistical morphs for three agglutinative languages and describe three different speech recognition systems that successfully utilize the n-gram language models trained for these units in the corresponding LVCSR tasks.

2 Building the lexicon and language models

2.1 Unsupervised discovery of morph units

Naturally, there are many ways to split the words into smaller units to reduce a lexicon to a tractable size. However, for a subword lexicon suitable for language modeling applications such as speech recognition, several properties are desirable:

1. The size of the lexicon should be small enough that the n-gram modeling becomes more feasible than the conventional word based modeling.
2. The coverage of the target language by words that can be built by concatenating the units should be high enough to avoid the out-of-vocabulary problem.
3. The units should be somehow meaningful, so that the previously observed units can help in predicting the next one.
4. In speech recognition one should be able to determine the pronunciation for each unit.

A common approach to find the subword units is to program the language-dependent grammatical rules into a morphological analyzer and utilize that to then split the text corpus into morphemes as in e.g. (Hirsimäki et al., 2005; Alumäe, 2005; Hacioglu et al., 2003). There are some problems related to ambiguous splits and pronunciations of very short inflection-type units, but also the coverage in, e.g., news texts may be poor because of many names and foreign words.

In this paper we have adopted a similar approach as (Hirsimäki et al., 2005). We use unsupervised learning to find the best units according to some cost function. In the Morfessor algorithm the minimized cost is the coding length of the lexicon and the words in the corpus represented by the units of the lexicon. This minimum description length based cost function is especially appealing, because it tends to give units that are both as frequent and as long as possible to suit well for both training the language models and also decoding of the speech. Full coverage of the language is also guaranteed by splitting the rare words into very short units, even to single phonemes if necessary. For language models utilized in speech

recognition, the lexicon of the statistical morphs can be further reduced by omitting the rare words from the input of the Morfessor algorithm. This operation does not reduce the coverage of the lexicon, because it just splits the rare words then into smaller units, but the smaller lexicon may offer a remarkable speed up of the recognition.

The pronunciation of, especially, the short units may be ambiguous and may cause severe problems in languages like English, in which the pronunciations can not be adequately determined from the orthography. In most agglutinative languages, such as Finnish, Estonian and Turkish, rather simple letter-to-phoneme rules are, however, sufficient for most cases.

2.2 Building the lexicon for open vocabulary

The whole training text corpus is first passed through a word splitting transformation as in Figure 1. Based on the learned subword unit lexicon, the best split for each word is determined by performing a Viterbi search with the unigram probabilities of the units. At this point the word break symbols are added between each word in order to incorporate that information in the statistical language models, as well. Then the n-gram models are trained similarly as if the language units were words including word and sentence break symbols as additional units.

2.3 Building the n-gram model over morphs

Even though the required morph lexicon is much smaller than the lexicon for the corresponding word n-gram estimation, the data sparsity problem is still important. Interpolated Kneser-Ney smoothing is utilized to tune the language model probabilities in the same way as found best for the word n-grams. The n-grams that are not very useful for modeling the language can be discarded from the model in order to keep the model size down. For Turkish, we used the entropy based pruning (Stolcke, 1998), where the n-grams, that change the model entropy less than a given treshold, are discarded from the model. For Finnish and Estonian, we used n-gram growing (Siivola and Pellom, 2005). The n-grams that increase the training set likelihood enough with respect to the corresponding increase in the model size are accepted into the model (as in the minimum description length principle). After the growing process the model is further pruned with entropy based pruning. The method allows us to train models with higher order n-grams, since the memory consumption is lower and also gives somewhat better models. Both methods can also be viewed as choosing the correct model complexity for the training data to avoid over-learning.

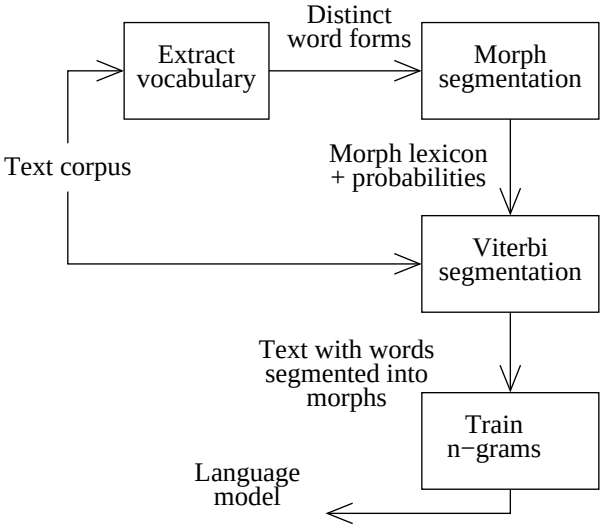


Figure 1: The steps in the process of estimating a language model based on statistical morphs from a text corpus (Hirsimäki et al., 2005).

3 Statistical properties of Finnish, Estonian and Turkish

Before presenting the speech recognition results, some statistical properties are presented for the three agglutinative languages studied. If we consider choosing a vocabulary of the 50k-70k most common words, as usual in English broadcast news LVCSR systems, the out-of-vocabulary (OOV) rate in English is typically smaller than 1%. Using the language model training data the following OOV rates can be found for a vocabulary including only the most common words: 15% OOV for 69k in Finnish (Hirsimäki et al., 2005), 10% for 60k in Estonian and 9% for 50k in Turkish. As shown in (Hacioglu et al., 2003) this does not only mean the same amount of extra speech recognition errors, but even more, because the recognizer tends to lose track when unknown words get mapped to those that are in the vocabulary. Even doubling the vocabulary is not a suf-

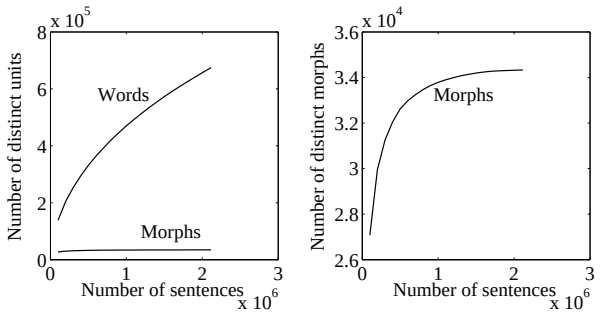


Figure 2: Vocabulary growth of words and morphs for Turkish language

ficient solution, because a vocabulary twice as large (120k) would only reduce the OOV rate to 6% in Estonian and 5% in Turkish. In Finnish even a 400k vocabulary of the most common words still gives 5% OOV in the language model training material.

Figure 2 illustrates the vocabulary explosion encountered when using words and how using morphs avoids this problem for Turkish. The figure on the left shows the vocabulary growth for both words and morphs. The figure on the right shows the graph for morphs in more detail. As seen in the figure, the number of new words encountered continues to increase as the corpus size gets larger whereas the number of new morphs encountered levels off.

4 Speech recognition experiments

4.1 About selection of the recognition tasks

In this work the morph-based language models have been applied in speech recognition for three different agglutinative languages, Finnish, Estonian and Turkish. The recognition tasks are speaker dependent and independent fluent dictation of sentences taken from newspapers and books, which typically require very large vocabulary language models.

4.2 Finnish

Finnish is a highly inflected language, in which words are formed mainly by agglutination and compounding. Finnish is also the language for which the algorithm for the unsupervised morpheme discovery (Creutz and Lagus, 2002) was originally developed. The units of the morph lexicon for the experiments in this paper were learned from a joint corpus containing newspapers, books and newswire stories of totally about 150 million words (CSC, 2001). We obtained a lexicon of 25k morphs by feeding the learning algorithm with the word list containing the 160k most common words. For language model training we used the same text corpus and the recently developed growing n-gram training algorithm (Siivola and Pellom, 2005). The amount of resulted n-grams are listed in Table 4. The average length of a morph is such that a word corresponds to 2.52 morphs including a word break symbol.

The speech recognition task consisted of a book read aloud by one female speaker as in (Hirsimäki et al., 2005). Speaker dependent cross-word triphone models were trained using the first 12 hours of data and evaluated by the last 27 minutes. The models included tied state hidden Markov models (HMMs) of totally 1500 different states, 8 Gaussian mixtures (GMMs) per state, short-time mel-cepstral features (MFCCs), maximum likelihood linear transformation (MLLT) and explicit phone duration models (Pylkkönen and Kurimo, 2004). The real-time factor of recognition speed was less than 10 xRT with a 2.2 GHz CPU. However, with the efficient LVCSR decoder utilized (Pylkkönen, 2005) it seems that by making an even smaller morph lexicon, such as 10k, the decoding speed could be optimized to only a few times real-time without an excessive trade-off with recognition performance.

4.3 Estonian

Estonian is closely related to Finnish and a similar language modeling approach was directly applied to the Estonian recognition task. The text corpus used to learn the morph units and train the statistical language model consisted of newspapers and books, altogether about 55 million words (Segakorpus, 2005). At first, 45k morph units were obtained as the best subword unit set from the list of the 470k most common words in the corpora. For speeding up the recognition, the morph lexicon was afterwards reduced to 37k by splitting the rarest morphs (occurring in only one or two words) further into smaller ones. Corresponding growing n-gram language models as in Finnish were trained from the Estonian corpora resulting the n-grams in Table 4.

The speech recognition task in Estonian consisted of long sentences read by 50 randomly picked held-out test speakers, 7 sentences each (a part of (Meister

et al., 2002)). Unlike the Finnish and Turkish microphone data, this data was recorded from telephone, i.e. 8 kHz sampling rate and narrow band data instead of 16 kHz and normal (full) bandwidth. The phoneme models were trained for speaker independent recognition using windowed cepstral mean subtraction and significantly more data (over 200 hours and 1300 speakers) than for the Finnish task. The speaker independence, together with the telephone quality and occasional background noises, made this task still a considerably more difficult one. Otherwise the acoustic models were similar cross-word triphone GMM-HMMs with MFCC features, MLLT transformation and the explicit phone duration modeling, except larger: 5100 different states and 16 GMMs per state. Thus, the recognition speed is also slower than in Finnish, about 20 xRT (2.2GHz CPU).

4.4 Turkish

Turkish is another a highly-inflected and agglutinative language with relatively free word order. The same Morfessor tool (Creutz and Lagus, 2005) as in Finnish and Estonian was applied to Turkish texts as well. Using the 360k most common words from the training corpus, 34k morph units were obtained. The training corpus consists of approximately 27M words taken from literature, law, politics, social sciences, popular science, information technology, medicine, newspapers, magazines and sports news. N-gram language models for different orders with interpolated Kneser-Ney smoothing as well as entropy based pruning were built for this morph lexicon using the SRILM toolkit (Stolcke, 2002). The number of n-grams for the highest order we tried (6-grams without entropy-based pruning) are reported in Table 4. In average, there are 2.37 morphs per word including the word break symbol.

The recognition task in Turkish consisted of approximately one hour of newspaper sentences read by one female speaker. We used decision-tree state clustered cross-word triphone models with approximately 5000 HMM states. Instead of using letter to phoneme rules, the acoustic models were based directly on letters. Each state of the speaker independent HMMs had a GMM with 6 mixture components. The HTK frontend (Young et al., 2002) was used to get the MFCC based acoustic features. The explicit phone duration models were not applied. The training data contained 17 hours of speech from over 250 speakers. Instead of the LVCSR decoder used in Finnish and Estonian (Pylkkönen, 2005), the Turkish evaluation was performed using another decoder (AT&T, 2003), Using a 3.6GHz CPU, the real-time factor was around one.

5 Results

The recognition results for the three different tasks: Finnish, Estonian and Turkish, are provided in Tables 1 – 3. In each task the word error rate (WER) and letter error rate (LER) statistics for the morph-based system is compared to a corresponding word-based system. The resulting morpheme strings are glued to words according to the word break symbols included in the language model (see Section 2.2) and the WER is computed as the sum of substituted, inserted and deleted words divided by the correct number of words. LER is included here as well, because although WER is a more common measure, it is not comparable between languages. For example, in agglutinative languages the words are long and contain a variable amount of morphemes. Thus, any incorrect prefix or suffix would make the whole word incorrect. The n-gram language model statistics are given in Table 4.

Finnish	lexicon	WER	LER
Words	400k	8.5	1.20
Morphs	25k	7.0	0.95

Table 1: The LVCSR performance for the speaker-dependent Finnish task consisting of book-reading (see Section 4.2). For a reference (word-based) language model a 400k lexicon was chosen.

Estonian	lexicon	WER	LER
Words	60k	56.3	22.4
Morphs	37k	47.6	18.9

Table 2: The LVCSR performance for the speaker-independent Estonian task consisting of read sentences recorded via telephone (see Section 4.3). For a reference (word-based) language model a 60k lexicon was used here.

Turkish	lexicon	WER	LER
Words			
3-gram	50k	38.8	15.2
Morphs			
3-gram	34k	39.2	14.8
4-gram	34k	35.0	13.1
5-gram	34k	33.9	12.4
Morphs, rescored by morph 6-gram			
3-gram	34k	33.8	12.4
4-gram	34k	33.2	12.3
5-gram	34k	33.3	12.2

Table 3: The LVCSR performance for the speaker-independent Turkish task consisting of read newspaper sentences (see Section 4.4). For the reference 50k (word-based) language model the accuracy given by 4 and 5-grams did not improve from that of 3-grams.

#	morph-based models		
ngrams	Finnish	Estonian	Turkish
1grams	24,833	37,061	34,332
2grams	2,188,476	1,050,127	655,621
3grams	17,064,072	7,133,902	1,936,263
4grams	25,200,308	8,201,543	3,824,362
5grams	7,167,021	3,298,429	4,857,125
6grams	624,832	691,899	5,523,922
7grams	23,851	55,363	-
8grams	0	1045	-
Sum	52,293,393	20,469,369	16,831,625

Table 4: The amount of different n-grams in each language model based on statistical morphs. Note that the Turkish language model was not prepared by the growing n-gram algorithm as the others and the model was limited to 6-grams.

In the Turkish recognizer the memory constraints during network optimization (Allauzen et al., 2004) allowed the use of language models only up to 5-grams. The language model pruning thresholds were optimized over a range of values and the best results are shown in Table 3. We also tried the same experiments with two-pass recognition. In the first pass, instead of the best path, lattice output was generated with the same language models with pruning. Then these lattices were rescored using the non-pruned 6-gram language models (see Table 4) and the best path was taken as the recognition output. For the word-based reference model, the two-pass recognition gave no improvements. It is likely that the language model training corpus was too small to train proper 6-gram word models. However, for the morph-based model, we obtained a slight improvement (0.7 % absolute) by two-pass recognition.

6 Discussion

The key result of this paper is that we can successfully apply the unsupervised statistical morphs in large vocabulary language models in all the three experimented agglutinative languages. Furthermore, the results show that in all the different LVCSR tasks, the morph-based language models perform very well and constantly dominate the reference language model based on words. The way that the lexicon is built from the word fragments allows the construction of statistical language models, in practice, for almost an unlimited vocabulary by a lexicon that still has a convenient size.

The recognition was here restricted to agglutinative languages and tasks in which the language used is both rather general and matches fairly well with the available training texts. Significant performance variation in different languages can be observed here, because of the different tasks and the fact that comparable recognition conditions and training resources have not been possible to arrange. However, we believe that the tasks are still both difficult and realistic enough to illustrate the difference of performance when using language models based on a lexicon of morphs vs. words in each task. There are no directly comparable previous LVCSR results on the same tasks and data, but the closest ones which can be found are slightly over 20% WER for the Finnish task (Hirsimäki et al., 2005), slightly over 40 % WER for the Estonian task (Alumäe, 2005) and slightly over 30 % WER for the Turkish task (Erdogan et al., 2005).

Naturally, it is also possible to prepare a huge lexicon and still succeed in recognition fairly well (Saraclar et al., 2002; McTait and Adda-Decker, 2003; Hirsimäki et al., 2005), but this is not a very convenient approach because of the resulting huge language models or the heavy pruning required to keep

them still tractable. The word-based language models that were constructed in this paper as reference models were trained as much as possible in the same way as the corresponding morph language models. For Finnish and Estonian the growing n-grams (Siivola and Pellom, 2005) were used including the option of constructing the OOV words from phonemes as in (Hirsimäki et al., 2005). For Turkish a conventional n-gram was built by SRILM similarly as for the morphs. The recognition approach taken for Turkish involves a static decoding network construction and optimization resulting in near real time decoding. However, the memory requirements of network optimization becomes prohibitive for large lexicon and language models as presented in this paper.

In this paper the recognition speed was not a major concern, but from the application point of view that is a very important factor to be taken into a account in the comparison. It seems that the major factors that make the recognition slower are short lexical units, large lexicon and language models and the amount of Gaussian mixtures in the acoustic model.

7 Conclusions

This work presents statistical language models trained on different agglutinative languages utilizing a lexicon based on the recently proposed unsupervised statistical morphs. To our knowledge this is the first work in which similarly developed subword unit lexica are developed and successfully evaluated in three different LVCSR systems in different languages. In each case the morph-based approach constantly shows a significant improvement over a conventional word-based LVCSR language models. Future work will be the further development of also the grammatical morph-based language models and comparison of that to the current approach, as well as extending this evaluation work to new languages.

8 Acknowledgments

We thank the Finnish Federation of the Visually Impaired for providing the Finnish speech data and the Finnish news agency (STT) and the Finnish IT center for science (CSC) for the text data. Our work was supported by the Academy of Finland in the projects *New information processing principles, Adaptive Informatics* and *New adaptive and learning methods in speech recognition*. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. The authors would like to thank Sabanci and ODTU universities for the Turkish acoustic and text data and AT&T Labs – Research for the software. This research is partially supported by SIMILAR Network of Excellence and TUBITAK BDP (Unified Doctorate Program of the Scientific and Technological Research Council of Turkey).

References

Cyril Allauzen, Mehryar Mohri, Michael Riley, and Brian Roark. 2004. A generalized construction of integrated speech recognition transducers. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Montreal, Canada.

Tanel Alumäe. 2005. Phonological and morphological modeling in large vocabulary continuous Estonian speech recognition system. In *Proceedings of Second Baltic Conference on Human Language Technologies*, pages 89–94.

Mehryar Mohri and Michael D. Riley. DCD Library – Speech Recognition Decoder Library. AT&T Labs – Research. `http://www.research.att.com/sw/tools/dcd/`.

Ebru Arisoy and Levent Arslan. 2005. Turkish dictation system for broadcast news applications. In *13th European Signal Processing Conference - EUSIPCO 2005*, Antalya, Turkey, September.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the Workshop on Morphological and Phonological Learning of ACL-02*, pages 21–30.

Mathias Creutz and Krista Lagus. 2005. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor. Technical Report A81, Publications in Computer and Information Science, Helsinki University of Technology. URL: `http://www.cis.hut.fi/projects/morpho/`.

J. Garofolo, G. Auzanne, and E. Voorhees. 2000. The TREC spoken document retrieval track: A success story. In *Proceedings of Content Based Multimedia Information Access Conference*, April 12-14.

P. Geutner, M. Finke, and P. Scheytt. 1998. Adaptive vocabularies for transcribing multilingual broadcast news. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Seattle, WA, USA, May.

H. Erdogan, O. Buyuk, K. Oflazer. 2005. Incorporating language constraints in sub-word based speech recognition. IEEE Automatic Speech Recognition and Understanding Workshop, Cancun, Mexico.

Kadri Hacioglu, Brian Pellom, Tolga Ciloglu, Ozlem Ozturk, Mikko Kurimo, and Mathias Creutz. 2003. On lexicon creation for Turkish LVCSR. In *Proceedings of 8th European Conference on Speech Communication and Technology*, pages 1165–1168.

Teemu Hirsimäki, Mathias Creutz, Vesa Siivola, Mikko Kurimo, Sami Virpioja, and Janne Pylkkönen. 2005. Unlimited vocabulary speech recognition with morph language models applied to Finnish. *Computer Speech and Language*. (accepted for publication).

Jan Kneissler and Dietrich Klakow. 2001. Speech recognition for huge vocabularies by using optimized subword units. In *Proceedings of the 7th European Conference on Speech Communication and Technology (Eurospeech)*, pages 69–72, Aalborg, Denmark.

CSC Tieteellinen laskenta Oy. 2001. Finnish Language Text Bank: Corpora Books, Newspapers, Magazines and Other. `http://www.csc.fi/kielipankki/`.

Kevin McTait and Martine Adda-Decker. 2003. The 300k LIMSI German Broadcast News Transcription System. In *Proceedings of 8th European Conference on Speech Communication and Technology*.

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*, pages 21–26.

NIST. 2000. *Proceedings of DARPA workshop on Automatic Transcription of Broadcast News*. NIST, Washington DC, May.

Janne Pylkkönen. 2005. New pruning criteria for efficient decoding. In *Proceedings of 9th European Conference on Speech Communication and Technology*.

Janne Pylkkönen and Mikko Kurimo. 2004. Duration modeling techniques for continuous speech recognition. In *Proceedings of the International Conference on Spoken Language Processing*.

Murat Saraclar, Michael Riley, Enrico Bocchieri, and Vincent Goffin. 2002. Towards automatic closed captioning: Low latency real time broadcast news transcription. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, Denver, CO, USA.

Segakorpus – Mixed Corpus of Estonian. Tartu University. `http://test.cl.ut.ee/korpused/segakorpus/`.

Vesa Siivola and Bryan Pellom. 2005. Growing an n-gram language model. In *Proceedings of 9th European Conference on Speech Communication and Technology*.

Vesa Siivola, Mikko Kurimo, and Krista Lagus. 2001. Large vocabulary statistical language modeling for continuous speech recognition. In *Proceedings of 7th European Conference on Speech Communication and Technology*, pages 737–747, Aalborg, Copenhagen.

Andreas Stolcke. 1998. Entropy-based pruning of backoff language models. In *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, pages 270–274.

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

Mate Szarvas and Sadaoki Furui. 2003. Evaluation of the stochastic morphosyntactic language model on a one million word Hungarian task. In *Proceedings of the 8th European Conference on Speech Communication and Technology (Eurospeech)*, pages 2297–2300.

S. Young, D. Ollason, V. Valtchev, and P. Woodland. 2002. The HTK book (for HTK version 3.2.), March.

Author Index